Hargittai - Kaszanyiczki

A VISUAL BASIC 4.0 programozási nyelv

WINDOWS 3.1x
WINDOWS 95
és WINDOWS NT alá

Hargittai Péter - Kaszanyiczki László

A VISUAL BASIC 4.0 programozási nyelv Windows 3.x, 95 és NT alá

Nyitott rendszerű képzés - Távoktatás -
Oktatási segédlete
Felsőoktatási tankönyv

LSI Oktatóközpont
A Mikroelektronika Alkalmazásának Kultúrájáért Alapítvány

Készült a Művelődési és Közoktatási Minisztérium támogatásával

Lektorálta: **Ribényi András**

ISBN 963 577 176 2

Kiadó: **LSI Oktatóközpont**
Felelős vezető: **Dr. Kovács Magda**
Témafelelős: **Flier István**

LITOVALENTIN KFT.

Tartalom

VISUAL BASIC 4.0 NYELVI ELEMEI 5

JELÖLÉSEK 6

A VISUAL BASIC 4.0 KULCSSZAVAK ÖSSZEFOGLALÓ TÁBLÁZATA 7

FÜGGVÉNYEK 27

UTASÍTÁSOK 81

METÓDUSOK 137

ESEMÉNYEK 222

TULAJDONSÁGOK 258

FORDÍTÁSI DIREKTÍVÁK 490

SQL NYELVI ELEMEK ÖSSZEFOGLALÁSA 492

FÜGGVÉNYEK 492

UTASÍTÁSOK 496

ÁLLÍTÁSOK 505

DEKLARÁCIÓK 506

MŰVELETEK 507

OPERÁTOROK 509

ZÁRADÉKOK 510

INDEX 519

Visual Basic 4.0 nyelvi elemei

A Visual Basic 4.0 verziója nem egyszerűen a Windows 95 megjelenése előtti verzió 32-bitesített változata, a rengeteg lényeges változás, újdonság miatt nyugodtan tekinthetjük minőségileg más programnak a korábbi verziókhoz képest.

A Microsoft már régóta tervezi, hogy az irodai alkalmazásaiban egységes makronyelvet vezet be, és ez a Visual Basic speciális verziója, az Application Edition lesz. Az Excel 5.0 verziójával ez meg is jelent. A Windows 95 alá megjelent újabb Microsoft irodai programcsomagok már valóban ennek a közös makronyelvnek a 2.0 verzióját tartalmazzák. A Visual Basic 4.0 is tartalmazza ezt, így a Visual Basicben fejlesztett kód egyszerűen átvihető az OLE Automatizmust támogató alkalmazások között.

A Visual Basic programfejlesztő rendszer 4.0 változatában az objektumokon, objektumgyűjteményeken és vezérlőelemeken értelmezett nyelvi elemek a következő alapvető csoportokba tartozhatnak:

függvény	(*Function*)
utasítás	(*Statement*)
metódus	(*Method*)
esemény	(*Event*)
tulajdonság	(*Property*)

A nyelvi elemek írásmódja, szintaktikája teljesen azonos az előző változatoknál megismertekkel.

A következőkben ennek a csoportosításnak megfelelően az egyes kulcsszavak funkcióját és használatát ismertetjük. A Visual Basic 4.0-s verzióban az eddigi, szűkösnek igazán nem nevezhető készlethez képest tovább növelték a programozási nyelv elemeinek számát, így a program lehetőségeinek kihasználásához most összességében több mint 860 kulcsszó használatát kell ismernünk. Feltételezzük, hogy a **Visual Basic 4.0-s** verzióját használók már tisztában vannak az egyes elemcsoportok általános jellemzőivel és felhasználási módjaival, ezért a korlátozott terjedelemre való tekintettel bővebben csak a **Visual Basic 3.0** című, az LSI Oktatóközpont által kiadott könyvben még nem szereplő vagy az azokhoz képest módosított hatású elemek használatához szükséges tudnivalókra térünk ki. A régebbi verziókból már ismert elemeket csak röviden, esetenként a korábbiakkal való azonosságra is hivatkozva ismertetjük. A csak kompatibilitási okokból megmaradt elemeket ugyan szerepeltetjük a könyvben, azonban szintaxisuk feltüntetését és használatuk bővebb leírását szükségtelennek tartottuk. A legfontosabbak esetek kivételével ugyancsak elhagytuk az egyes nyelvi elemek használatát bemutató mintapéldákat, ezek részint a programozási ismereteket összefoglaló könyvben, részint a Visual Basic 4.0 súgójában találhatók meg.

A könyv végén külön egységet képeznek az SQL elemek használatára vonatkozó információk.

Jelölések

Ezen könyvben a funkcionális szövegrészek megkülönböztetését és kiemelését több betűtípus használata segíti. Ezek a betűtípusok a következők:

- Az alaptípus az, amelyikkel a folyamatos ismertető, magyarázó szöveg jelenik meg.
- A **Visual Basic kulcsszavait** és az **előredefiniált szimbolikus konstansokat** ennek a típusnak a vastagított változatával emeltük ki. Csak nagybetűsek, de nem vastagítottak a FÁJLNEVEK és NÉVKITERJESZTÉSEK.
- Az argumentumlistában elfoglalt pozíciójukkal megkülönböztetett *objektumazonosítókat és argumentumokat* dőlt betűkkel jelöltük. A vastagított ***argumentum*** azt jelzi, hogy arra az adott névvel lehet hivatkozni, az argumentumlistán belüli pozíciója közömbös.
- A képernyőn a programablakok és párbeszédpanelek menüi, menüparancsai, címkéi és nyomógombfeliratai ezzel a betűtípussal jelennek meg, megkülönböztetésül az esetlegesen azonos, de más funkciójú szövegektől.
- A `billentyűket` és `billentyűkombinációkat` valamint a `programkód-mintákat` az írógépszerű betűtípussal jelöltük.

A nyelvi elemek használata ismertetésekor speciális karakterek is előfordulnak, így a zárójelek , a vonás (|) és az aláhúzás (_) karakterek és a három pont (...). Ezek jelentése:

( ) argumentumlistát jelöl, azaz a függvények, utasítások használatához kötelező és feltételes argumentumok felsorolását tartalmazza,

[ ] feltételességet jelöl, azaz a szögletes zárójelben szereplő elem megadása nem kötelező,

{ | } vagylagosságot jelöl, azaz az argumentumlista kapcsos zárójelei között a vonással elválasztott elemek közül az adott környezetben csak az egyik szerepelhet,

_ folytonosságot jelöl, azaz az egymás utáni következő szavak, utasítások egyetlen logikailag sorba tartoznak, és végül a

... ismétlést jelöl, azaz az előtte szereplő elemekből álló egység többször is használható.

A Visual Basic 4.0 kulcsszavak összefoglaló táblázata

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
#Const					
#If...Then...Else					
Abs	*				
AbsolutePosition					*
Action					*
Activate				*	
ActiveControl					*
ActiveForm					*
Add			*		
AddItem			*		
AddNew			*		
AfterColUpdate				*	
AfterDelete				*	
AfterInsert				*	
AfterUpdate				*	
Align					*
Alignment					*
AllowAddNew					*
AllowDelete					*
AllowRowSizing					*
AllowSizing					*
AllowUpdate					*
AllowZeroLength					*
AllPermission					*
ALTER TABLE		*			
AppActivate		*			
Appearance					*
Append			*		
AppendChunk			*		
AppIsRunning					*
Active					*
Arrange			*		
Array	*				
Asc	*				
AscB	*				
Atn	*				

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
Attributes					*
AutoActivate					*
AutoRedraw					*
AutoShowChildren					*
AutoSize					*
AutoVerbMenu					*
Avg	*				
BackColor					*
BackStyle					*
Beep		*			
BeforeColUpdate				*	
BeforeDelete				*	
BeforeInsert				*	
BeforeUpdate				*	
BeginTrans			*		
BOF					*
BOFAction					*
Bold					*
Bookmark					*
Bookmarkable					*
BorderColor					*
BorderStyle					*
BorderWidth					*
BoundColumn					*
BoundText					*
CacheSize					*
Call		*			
Cancel					*
CancelError					*
CancelUpdate			*		
Caption					*
CBool	*				
CByte	*				
CCur	*				
CDate	*				
CDbl	*				
CellSelected					*
CellText			*		

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
CellValue			*		*
CentimeterToPoints			*		
Change				*	
ChDir		*			
ChDrive		*			
Checked					*
Choose	*		*		
Chr	*				
ChrB	*				
CInt	*				
Circle			*		
Class					*
Clear			*		
Click				*	
Clip					*
ClipControls					*
CLng	*				
Clone			*		
Close		*	*		
Cls			*		
Clustered					*
Col					*
ColAlignment					*
ColContaining			*		
ColIndex					*
ColIsVisible					*
CollatingOrder					*
Color					*
ColorMode					*
ColPos					*
ColResize				*	
Cols					*
ColumnHeaders					*
Columns					*
ColWidth					*
Command	*				
Comments					*
CommitTrans			*		

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
CompactDatabase			*		
CompanyName					*
ConflictTable					*
Connect					*
Const		*			
Container					*
ControlBox					*
Copies					*
Copy			*		*
CopyQueryDef			*		*
Cos	*				
Count	*				*
CREATE INDEX		*			
CREATE TABLE		*			
CreateDatabase	*		*		
CreateDynaset			*		
CreateEmbed			*		
CreateField			*		
CreateGroup			*		
CreateIndex			*		
CreateLink			*		
CreateObject			*		
CreateProperty			*		
CreateQueryDef			*		
CreateRelation			*		
CreateSnapshot			*		
CreateTableDef			*		
CreateUser			*		
CreateWorkspace			*		
CSng	*				
CStr	*				
CurDir	*				
CurrentX					*
CurrentY					*
CVar	*				
CVErr	*				
Data					*
Database					*

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
DatabaseName					*
DataChanged					*
DataField					*
DataSource					*
DataText					*
DataUpdatable					*
Date	*	*			
DateAdd	*				
DateCreated					*
DateDiff	*				
DatePart	*				
DateSerial	*				
DateValue	*				
Day	*				
DblClick				*	
DDB	*				
Deactivate				*	
Declare		*			
Default					*
DefaultExt					*
DefaultUser					*
DefaultValue					*
DefColWidth					*
DefaultPassword					*
DefBool		*			
DefByte					
DefCur		*			
DefDate		*			
DefDbl		*			
DefInt		*			
DefLng		*			
DefObj		*			
DefSng		*			
DefStr		*			
DefVar		*			
Delete			*		
DELETE		*			
DeleteQueryDef			*		

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
Description					*
DesignMasterID					*
DeviceName					*
DialogTitle					*
Dim		*			
Dir	*				
DisplayType					*
DistinctCount					*
DividerStyle					*
Do...Loop		*			
DoEvents	*				
DoVerb			*		
Drag			*		
DragDrop				*	
DragIcon					*
DragMode					*
DragOver				*	
DrawMode					*
DrawStyle					*
DrawWidth					*
Drive					*
DriverName					*
DROP		*			
DropDown				*	
Duplex					*
Edit			*		
EditMode					*
Enabled					*
End		*			
EndDoc			*		*
Environ	*				
EOF	*				*
EOFAction					*
Erase		*			
Error	*			*	
Exclusive					*
Execute			*		*
ExecuteSQL			*		

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
EXEName					*
Exit		*			
Exp	*				
FetchVerbs			*		
Fields					*
FieldSize			*		
FileAttr	*				
FileCopy		*			
FileDateTime	*				
FileDescription					*
FileLen	*				
FileName					*
FileNumber					*
FileTitle					*
FileCache			*		
FillColor					*
FillStyle					*
Filter					*
FilterIndex					*
FindFirst			*		
FindLast			*		
FindNext			*		
FindPrevious			*		
First	*				
FirstRow					*
Fix	*				
FixedAlignment					*
FixedCols					*
FixedRows					*
Flags					*
Font					*
FontBold					*
FontCount					*
FontItalic					*
FontName					*
Fonts					*
FontSize					*
FontStrikethru					*

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
FontTransparent					*
FontUnderline					*
For Each...Next		*			
For...Next		*			
ForeColor					*
Foreign					*
ForeignName					*
ForeignTable					*
Format	*				*
FreeFile	*				
FreeLocks		*			
FromPage					*
Function		*			
FV	*				
Get		*			
GetAllSettings	*				
GetAttr	*				
GetBookmark			*		
GetChunk			*		
GetData			*		
GetFormat			*		
GetObject	*				
GetRows			*		
GetSetting			*		
GetText			*		
GoSub...Return		*			
GotFocus				*	
GoTo		*			
GridLines					*
GridLineWidth					*
Handle					*
hDC					*
HeadBackColor					*
HeadClick				*	
HeadFont					*
HeadForeColor					*
HeadLines					*
Height					*

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
HelpCommand					*
HelpContext					*
HelpContextID					*
HelpFile					*
HelpKey					*
Hex	*				
Hide			*		
Hidden					*
HiddeSelection					*
HighLight					*
hInstance					*
HostName					*
Hour	*				
hPal					*
hWnd					*
Icon					*
Idle			*		
If...Then...Else		*			
IgnoreNulls					*
IIf	*				
Image					*
IMEStatus	*				
Index					*
Inherit					*
IniPath					*
InitDir					*
Initialize				*	
Input#		*			
Input	*				
InputB	*				
InputBox	*				
INSERT INTO		*			
Insert			*		
InsertObjDlg			*		
Instancing					*
InStr	*				
InStrB	*				
Int	*				

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
IntegralHeight					*
Interval					*
IPmt	*				
IRR	*				
IsArray	*				
IsDate	*				
IsEmpty	*				
IsError	*				
IsMissing	*				
IsNull	*				
IsNumeric	*				
IsObject	*				
IsolateODBCTrans					*
Italic					*
Item			*		
ItemData					*
KeepLocal					*
KeyDown			*		
KeyPress			*		
KeyPreview					*
KeyUp			*		
Kill		*			
KillDoc			*		
LargeChange					*
LastDLLError					*
LastModified					*
LastUpdated					*
Last	*				
LBound	*				*
LCase	*				
Left	*				*
LeftB	*				
LeftCol					*
LegalCopyright					*
LegalTrademarks					*
Len	*				
LenB	*				
Let		*			

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
Line Input #		*			
Line			*		
LinkClose				*	
LinkError				*	
LinkExecute			*	*	
LinkItem					*
LinkMode					*
LinkNotify					*
LinkOpen				*	
LinkPoke			*		
LinkRequest			*		
LinkSend			*		
LinkTimeout					*
LinlTopic					*
List					*
ListCount					*
ListField					*
ListFields			*		
ListIndex					*
ListIndexes			*		
ListParameters			*		
ListTables			*		
Load		*		*	
LoadPicture	*				
LoadResData	*				
LoadResPicture	*				
LoadResString	*				
Loc	*				
Lock...Unlock		*			
Locked					*
LockEdits					*
LOF	*				
Log	*				
LoginTimeout					*
LogMessages					*
LostFocus				*	
lpOLEObject					*
LSet		*			

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
LTrim	*				
Major					*
MakeReplica			*		
MatchEntry					*
MatchWithList					*
Max					*
MaxButton					*
MaxFileSize					*
MaxLength					*
MDIChild					*
Mid	*	*			
MidB	*	*			
Min	*				
MinButton					*
Minor					*
Minute	*				
MIRR	*				
MiscFlags					*
MkDir		*			
Month	*				
MouseDown				*	
MouseIcon					*
MouseMove				*	
MousePointer					*
MouseUp				*	
Move			*		
MoveFirst			*		
MoveLast			*		
MoveNext			*		
MovePrevious			*		
MsgBox	*				
MultiLine					*
MultiSelect					*
Name		*			*
Negotiate					*
NegotiateMenus					*
NegotiatePosition					*
NegotiateToolbars					*

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
NewIndex					*
NewPage			*		
NewPassword			*		
NoMatch					*
Normal					*
Now	*				
NPer	*				
NPV	*				
NumberFormat					*
Object					*
ObjectAcceptFormats					*
ObjectAcceptFormatsCount					*
ObjectGetFormats					*
ObjectGetFormatsCount					*
ObjectMove				*	
ObjectVerbFlags					*
ObjectVerbs					*
ObjectVerbsCount					*
Oct	*				
ODBCTimeout					*
OLEDropAllowed					*
OLERequestPendingMsgText					*
OLERequestPendingMsgTitle					*
OLERequestPendingTimeout					*
OLEServerBusyMsgText					*
OLEServerBusyMsgTitle					*
OLEServerBusyRaiseError					*
OLEServerBusyTimeout					*
OLEType					*
OLETypeAllowed					*
On Error		*			
On...GoSub		*			
On...GoTo		*			
Open		*			
OpenDatabase	*		*		
OpenQueryDef			*		
OpenRecordset			*		
OpenTable			*		

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
Option Base		*			
Option Compare		*			
Option Explicit		*			
Option Private		*			
Options					*
OrdinalPosition					*
Orientation					*
Owner					*
Page					*
Paint				*	
PaintPicture			*		
PaperBin					*
PaperSize					*
Parent					*
Partition	*				
Password					*
PaswordChar					*
Paste			*		
PasteOK					*
PasteSpecialDlg			*		
Path					*
PathChange				*	
Pattern					*
PatternChange				*	
PercentPosition					*
Permissions					*
Picture					*
PID					*
Pmt	*				
Point			*		
PopupMenu			*		
Port					*
PPmt	*				
PrevInstance					*
Primary					*
Print #		*			
Print			*		
PrinterDefault					*

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
PrintForm			*		
PrintQuality					*
Private		*			
ProductName					*
Property Get		*			
Property Let		*			
Property Set		*			
PSet			*		
Public		*			*
Put		*			
PV	*				
QBColor	*				
QueryTimeout					*
QueryUnload				*	
Raise			*		
Randomize		*			
Rate	*				
ReadFromFile			*		
ReadOnly					*
Rebind			*		
RecordCount					*
RecordsAffected					*
RecordSelectors					*
Recordset					*
RecordsetType					*
RecordSource					*
ReDim		*			
Refill			*		
Refresh			*		
RefreshLink			*		
RegisterDatabase		*	*		
Rem		*			
Remove			*		
RemoveItem			*		
RepairDatabase		*	*		
Replicable					*
ReplicaID					*
Reposition				*	

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
Requery			*		
Required					*
Reset		*			
Resize				*	
Restartable					*
Resume		*			
ReturnsRecords					*
Revision					*
RGB	*				
Right	*				
RightB	*				
RmDir		*			
Rnd	*				
Rollback		*	*		
Row					*
Rows					*
RowBookmark			*		
RowColChange				*	
RowContaining			*		
RowDividerStyle					*
RowHeight					*
RowIsVisible					*
RowLoaded				*	
RowPos					*
RowResize				*	
RowSource					*
RowTop			*		
RSet		*			
RTrim	*				
SavePicture		*			
SaveSetting		*			
SaveToFile			*		
SaveToOle1File			*		
Scale			*		
ScaleHeight					*
ScaleWidth					*
ScaleX			*		
ScaleY			*		

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
ScaleLeft					*
ScaleTop					*
ScaleMode					*
Scroll			*	*	
ScrollBars					*
Second	*				
Seek	*	*	*		
SelBookmarks					*
SelChange				*	
SelCount					*
SELECT		*			
Select Case		*			
SELECT...INTO		*			
Selected					*
SelectedItem					*
SelEndCol					*
SelEndRow					*
SelLength					*
SelStart					
SelStartCol					*
SelStartRow					*
SelText					*
SendKeys		*			
Set		*			
SetAttr		*			
SetData			*		
SetDataAccessOption		*			
SetDefaultWorkspace		*			
SetFocus			*		
SetText			*		
Sgn	*				
Shape					*
Shell	*				
Shortcut					*
Show			*		
ShowColor			*		
ShowFont			*		
ShowHelp			*		

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
ShowInTaskbar					*
ShowOpen			*		
ShowPrinter			*		
ShowSave			*		
ShowWhatsThis			*		
Sin	*				
Size					*
SizeMode					*
SLN	*				
SmallChange					*
Sort					*
Sorted					*
Source					*
SourceDoc					*
SourceField					*
SourceItem					*
SourceTable					*
SourceTableName					*
Space	*				
Spc	*				
SQL					*
Sqr	*				
StartMode					*
Static		*			
StDev	*				
StDevP	*				*
Stop		*			
Str	*				
StrComp	*				
StrConv	*				
Stretch					*
StrikeTrough					*
String	*				
Style					*
Sub		*			
Sum	*				
Switch	*				
SYD	*				*

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
Synchronize			*		
System					*
Tab	*				
TabIndex					*
Table					*
TabStop					*
Tag					*
Tan	*				
TaskVisible					*
Terminate				*	
Text					*
TextHeight			*		
TextWidth			*		
Time	*	*			
Timer	*			*	
TimeSerial	*				
TimeValue	*				
Title					*
Top					*
TopIndex					*
ToPage					*
ToRow					*
TrackDefault					*
Transaction					*
TRANSFORM		*			
Trim	*				
TwipsPerPixelX					*
TwipsPerPixelY					*
Type		*			*
TypeName	*				
UBound	*				*
UCase	*				
Underline					*
Unique					*
Unload		*		*	
Unlock		*			
Updatable					*
Update			*		

Kulcsszó:	Függvény	Utasítás	Metódus	Esemény	Tulajdonság
UPDATE		*			
UpdateControls			*		
Updated				*	
UpdateOptions					*
UpdateRecord			*		
UseMnemonic					*
UserName					*
V1xNullBehavior					*
Val	*				
Validate				*	
ValidateOnSet					*
ValidationRule					*
ValidationText					*
Value					*
Var	*				
VarP	*				
VarType	*				
Verb					*
Version					*
Visible					*
VisibleCols					*
VisibleCount					*
VisibleItem					*
VisibleRows					*
Weekday	*				
Weight					*
WhatsThisButton					*
WhatsThisHelp					*
WhatsThisHelpID					*
WhatsThisMode				*	
While...Wend		*			
Width#		*			
Width					*
WindowList					*
WindowState					*
With		*			
WordWrap					*
Write#		*			

X1					*
X2					*
Y1					*
Y2					*
Year	*				
Zoom					*
ZOrder			*		

Függvények

Abs()

Szintaxis: **Abs(** *numerikus_kifejezés*)

A függvény a *numerikus_kifejezés* argumentumban szereplő bármely érvényes számnak vagy numerikus kifejezés értékének abszolút, előjel nélküli alakját adja vissza. Ha a *numerikus_kifejezés* értéke **Null**, az eredmény is **Null** lesz, míg az inicializálatlan változót tartalmazó *numerikus_kifejezés* esetén a függvény nulla értéket ad vissza.

Array ()

Szintaxis: **Array(** [*argumentumlista*])

A függvény eredménye az *argumentumlista* tetszőleges számú, egymástól vesszővel elválasztva megadott elemének megfelelő nagyságú tömböt tartalmazó **Variant**. A lista első eleme a tömb első elemébe kerül, a további elemek pedig rendre a következő tömbelemekbe. A **Variant** egyes elemei a tömbön belüli indexükre hivatkozva érhetők el. A tömb első elemének indexe (1 vagy 0) az **Option Base** utasítással állítható be. Üres ("") argumentumlista esetén az eredmény nulla hosszúságú, üres tömb lesz.

Az **Array( )** függvénnyel létrehozott **Variant** — a rögzített hosszúságú karakterláncokat és a felhasználó által definiált változókat tartalmazó tömbön kívül — bármely típusú tömböt tartalmazhat.

Asc( )

Szintaxis: **Asc(** *karakteres_kifejezés*)

A függvény a *karakteres_kifejezés* argumentumban adott bármely érvényes karakteres kifejezés első karakterének karakterkódját adja vissza. Üres ("") argumentum futásidejű hibát okoz.

AscB( )

Szintaxis: **AscB(** *karakteres_kifejezés*)

Az **AscB( )** az **Asc( )** függvényhez hasonló funkciójú, azonban az érvényes, **Byte** adatokat tartalmazó *karakteres_kifejezés* első byte-ját adja meg.

Atn( )

Szintaxis: **Atn(** *numerikus_kifejezés*)

A függvény a *numerikus_kifejezés* argumentumban adott bármely érvényes szám vagy numerikus kifejezés számszerű eredményét az arkusz tangens, azaz az inverz tangensfüggvény argumentumaként értelmezi, és az ehhez tartozó szög $-\pi/2$ és $\pi/2$ radián közötti értékét adja vissza. Az eredményt fokra az 1 radián = $180/\pi$ = 57,2957795130824 fok arányosság szerint lehet átszámítani.

CBool( )

Szintaxis: **CBool(** *kifejezés*)

A függvény a *kifejezés* argumentumban adott bármely érvényes numerikus vagy numerikus értékként értelmezhető karakteres kifejezés eredményét a következő szabály szerint alakítja **Boolean** típusúvá: ha a *kifejezés* nulla, a **CBool( )** függvény **False** logikai értéket ad vissza, egyébként az eredmény **True**. Numerikus adattípusúként nem értelmezhető *kifejezés* futásidejű hibát okoz.

CByte( )

Szintaxis: **CByte(** *kifejezés*)

A függvény a *kifejezés* argumentumban adott bármely érvényes numerikus vagy karakteres kifejezés eredményét **Byte** típusúvá alakítja. A **Byte** adattípus számára elfogadható tartományt (0-255) túllépő argumentumérték hibát okoz.

A függvény segítségével az alapértelmezettként **Currency**, **Single**, **Double**, **Long** vagy **Integer** aritmetika helyett **Byte** aritmetika alkalmazását írhatjuk elő.

A **Val( )** helyett a **CByte( )** függvényt használva a területi beállítástól függő bármely adattípus is helyesen alakítható **Byte** típusúvá, és így például egységes alakban kezelhetőkké válnak az eltérő számtagolási rendszerek.

CCur()

Szintaxis: **CCur(** *kifejezés* **)**

A függvény a *kifejezés* argumentumban megadott bármely érvényes numerikus vagy karakteres kifejezés eredményét **Currency** típusúvá alakítja. Az adattípus számára a -922337203685,4775808 és 922337203685,4775807 között érvényes tartományt túllépő argumentum hibát okoz.

A függvény segítségével az alapértelmezettként **Single**, **Double**, **Long** vagy **Integer** aritmetika helyett **Currency** aritmetika alkalmazását írhatjuk elő.

A **Val( )** helyett a **CCur( )** függvényt használva például a területi beállítástól függő bármely adattípus is helyesen alakítható **Currency** típusúvá, és így például egységes alakban kezelhetőkké válnak az eltérő számtagolási és pénzegység rendszerek.

CDate()

Szintaxis: **CDate(** *dátum* **)**

A függvény a *dátum* argumentumban megadott bármely érvényes dátumkifejezést **Date** típusúvá alakít.

A **CDate( )** függvény egyaránt elfogad a területi beállításnak megfelelő, érvényes alakú karakteres dátumkifejezésként és valós számként megadott dátum/idő értékeket: a számként megadott argumentum egész részét teljes egészében dátumnak, a tizedesjel utáni részét pedig a nap kezdetéhez (éjfélhez) viszonyított időpontnak tekinti. Karakteres kifejezésként például a "February 24, 1993" alak helyett ajánlottabb a #February 24, 1993# vagy #2/24/1993# alakú kifejezések használata. A hosszú alakú dátumot tartalmazó *dátum* karakterlánca nem tartalmazhatja a hét napjára vonatkozó részt. Egy adott kifejezés **Date** típusúvá konvertálhatóságát az **IsDate( )** függvény segítségével ellenőrizhetjük.

CDbl()

Szintaxis: **CDbl(** *kifejezés* **)**

A függvény a *kifejezés* argumentumban adott bármely érvényes numerikus vagy karakteres kifejezés eredményét **Double** típusúvá alakítja.

A függvény segítségével az alapértelmezettként **Currency**, **Single**, **Long** vagy **Integer** aritmetika helyett **Currency** aritmetika alkalmazását írhatjuk elő. Természetesen az átalakítás az eredeti kifejezés pontosságát nem javítja. A megengedett tartományon kívülre eső értékű argumentum hibajelzést okoz.

A **Val( )** helyett a **CDbl( )** függvényt használva a területi beállítástól függő bármely adattípus is helyesen alakítható **Double** típusúvá, és így például egységes alakban kezelhetőkké válnak az eltérő számtagolási rendszerek.

Choose( )

Szintaxis: **Choose(** *index*, *Variant_kifejezés1*, [*Variant_kifejezés2*], ...)

A függvény az argumentumlista **Variant** típusú kifejezéseket tartalmazó listájából az *index* értékével kijelölt elemet adja vissza, vagyis elsősorban a több lehetőséget tartalmazó listáról történő választásra használható. Az *index* szám, numerikus kifejezés vagy mező lehet. Ha a szám vagy numerikus kifejezés értéke nem egész szám, az eredmény a **Fix( )** függvény szabályai szerint kerekítődik. Az *index* értékének 1 és a *Variant_kifejezés* listáján szereplő bejegyzések száma között kell lennie, ellenkező esetben a függvény **Null** értéket ad vissza.

A függvény az *index* értékétől függetlenül az argumentumlistában szereplő **Variant** kifejezések mindegyikét kiértékeli, így ha azokban például egy-egy **MsgBox( )** függvény szerepel, egymás után az összes üzenetpanel megjelenik a képernyőn.

Chr( )

Szintaxis: **Chr(** *karakterkód*)

A függvény eredménye a *karakterkód* argumentumban megadott, 0 és 255 közötti egész számnak a kódtáblázatban megfelelő karaktere lesz. A függvényt a 0 és 31 közötti karakterkódú, nem nyomtatható ASCII karakterek karakterláncokba illesztésére használhatjuk.

ChrB( )

Szintaxis: **ChrB(** *karakterkód*)

A **ChrB( )** a **Chr( )** függvényhez hasonló funkciójú, azonban egyetlen, a *karakterkód* argumentumnak megfelelő byte-ot ad eredményként, és **Byte** típusú adatok karakterláncba illesztésére használhatjuk.

CInt( )

Szintaxis: **CInt(** *kifejezés*)

A függvény a *kifejezés* argumentumban adott bármely érvényes numerikus vagy karakteres kifejezés eredményét **Integer** típusúvá alakítja. A megengedett -32768 és 32767 közötti tartományon kívülre eső értékű argumentum hibajelzést okoz. Az eredmény átalakítás során szükségessé váló kerekítése nem a **Fix( )** vagy **Int( )** függvények szerint történik, itt a 0.5 végződésű számok eredménye mindig a legközelebbi páros szám lesz.

A függvény segítségével az alapértelmezettként **Currency**, **Single** vagy **Double** aritmetika helyett **Integer** aritmetika alkalmazását írhatjuk elő.

A **Val( )** helyett a **CInt( )** függvényt használva a területi beállítástól függő bármely

adattípus is helyesen alakítható **Integer** típusúvá, és így például egységes alakban kezelhetőkké válnak az eltérő számtagolási rendszerek.

CLng()

Szintaxis: **CLng(** *kifejezés* **)**

A függvény a *kifejezés* argumentumban adott bármely érvényes numerikus vagy karakteres kifejezés eredményét **Long** típusúvá alakítja. A megengedett -2147483648 és 2147483647 közötti tartományon kívül eső értékű kifejezés hibajelzést okoz. Az eredmény átalakítás során szükségessé váló kerekítése nem a **Fix( )** vagy **Int( )** függvények szerint történik, itt a 0.5 végződésű számok eredménye mindig a legközelebbi páros szám lesz.

A függvény segítségével az alapértelmezettként **Currency**, **Single** vagy **Double** aritmetika helyett **Long** aritmetika alkalmazását írhatjuk elő.

A **Val( )** helyett a **CLng( )** függvényt használva a területi beállítástól függő bármely adattípus is helyesen alakítható **Long** típusúvá, és így például egységes alakban kezelhetőkké válnak az eltérő számtagolási rendszerek.

Command

Szintaxis: **Command**

Ez az argumentum nélküli függvény a Visual Basic vagy az abban fejlesztett, futtatható alakra fordított (.EXE) program parancssorból történő indításakor a parancssornak a program számára átadandó (a /CMD utáni) argumentumrészt adja vissza kiolvasására. Az eredményt a kód ablakban az Options párbeszédpanel Advanced fülén keresztül lehet módosítani.

Cos()

Szintaxis: **Cos(** *szám* **)**

A függvény a *szám* argumentumban adott, egy szög radiánban kifejezett argumentumaként értelmezhető érvényes számhoz vagy numerikus kifejezés eredményéhez tartozó, -1 és 1 közé eső koszinusz értéket adja vissza. A fok radiánra történő átszámítása az 1 fok = $\pi/180$ = 0,0174532925199433 radián azonosság szerint történhet.

CreateDatabase

Szintaxis:

Ez az új Microsoft Jet adatbázist létrehozó, az adatbázist lemezre mentő, majd egy

új **Database** objektumot létrehozó, és az írásra/olvasásra megnyitott **Database** objektumot visszaadó függvény csak kompatibilitási okokból maradt meg. A **Workspace** objektummal a **CreateDatabase** metódus használata javasolt.

CreateObject()

Szintaxis: **CreateObject(** *osztály* **)**

Ennek az OLE Automatizmus objektum létrehozására alkalmazható függvénynek az *osztály* argumentumban az objektumot szolgáltató alkalmazás nevét és a létrehozandó objektum típusát vagy osztályát az *alkalmazásnév.objektumtípus* alakban kell megadni. Minden egyes, az OLE Automatizmus támogatására képes alkalmazás legalább egy objektumtípust is meghatároz. Az OLE Automatizmus objektum létrehozáshoz a függvény által visszaadott objektumot egy objektumváltozónak kell átadni. A függvény által visszaadott objektum argumentumként objektumot váró függvényben is megadható.

A **CreateObject( )** függvényt akkor lehet használni, ha az objektumnak még nem létezik példánya. Ha már van objektumpéldány vagy egy alkalmazást akarunk elindítani, a **GetObject( )** függvény használható. A magát egyetlen példányban deklaráló objektumra a **CreateObject( )** függvényt akárhányszor végrehajtva is csak egyetlen objektumpéldány jön létre.

CSng()

Szintaxis: **CSng(** *kifejezés* **)**

A függvény a *kifejezés* argumentumban adott bármely érvényes numerikus vagy karakteres kifejezés eredményét **Single** típusúvá alakítja. A **Single** adattípusra megengedett tartományon kívül eső értékű kifejezés hibajelzést okoz.

A függvény segítségével az egyébként **Currency**, **Double**, **Integer** vagy **Long** aritmetika helyett **Single** aritmetika alkalmazását írhatjuk elő.

A **Val( )** helyett a **CSng( )** függvényt használva például a területi beállítástól függő bármely adattípus is helyesen alakítható **Single** típusúvá, és ezzel egységes alakban kezelhetőkké válnak a különböző számtagolási rendszerek.

CStr()

Szintaxis: **CStr(** *kifejezés* **)**

A függvény a *kifejezés* argumentumban adott bármely érvényes numerikus vagy karakteres kifejezés eredményét **String** típusúvá alakítja. A függvény segítségével az egyébként számként megjelenő adatok karakterláncként kezelhetővé válnak. A függvény eredménye a *kifejezés* eredményének adattípusától függően a következő:

Ha a *kifejezés* típusa:	Az eredmény:
Boolean	**True** vagy **False** tartalmú **String**
Date	A rendszer beállításának megfelelő rövid alakú dátumot tartalmazó **String**
Null	Futásidejű hiba
Empty	Üres **String** ("")
Error	Az **Error** szót és egy azt követő hibaszámot tartalmazó **String**
Más numerikus típus	A számjegyeket tartalmazó **String**

A **Str( )** helyett a **Csng( )** függvényt használva a területi beállítástól függő bármely adattípus is helyesen alakítható **Single** típusúvá, és ezzel egységes alakban kezelhetőkké válnak a különböző számtagolási rendszerek.

CurDir( )

Szintaxis: **CurDir** [(*meghajtónév*)]

A függvény a *meghajtónév* argumentumban adott karaktersorozat első karakterével, mint azonosítóval meghatározott meghajtón az éppen érvényes elérési utat adja vissza. Hiányzó vagy üres karakterláncot ("") tartalmazó *meghajtónév* esetén a függvény eredménye az aktuális meghajtó elérési útja lesz. Nem létező vagy nem specifikált meghajtóra történő hivatkozás esetén, vagy a meghajtónév karaktersorozat első karaktereként nem betűt használva hibajelzést kapunk.

CVar( )

Szintaxis: **CVar(** *kifejezés* **)**

A függvény a *kifejezés* argumentumban megadott bármely érvényes numerikus vagy karakteres kifejezés eredményét **Variant** típusúvá alakítja.

CVDate( )

Szintaxis:

Noha kompatibilitási okokból a Visual Basic előző verzióiban használt **CVDate( )** függvény a **CDate( )** szintaxisával azonos módon továbbra is érvényes, belső **Date** típus hiányában nincs rá szükség. A **CVDate( )** függvény az aktuális **Date** típus helyett **Date** altípusú **Variant** értéket adott vissza, amelyet most — hasonlóképpen minden belső típus vele egyenértékű **Variant** altípusának konvertálásához — a *dátum* **Date** típussá konvertálásának eredményét egy **Variant** változónak átadva kaphatunk meg.

CVErr( )

Szintaxis: **CVErr(** *hibaszám* **)**

A függvény a *hibaszám* argumentumban megadott bármely érvényes hibaszámhoz a felhasználó által specifikált hibaszámot tartalmazó **Error** altípusú **Variant** értéket adja eredményül.

A függvény használatával a saját eljárásainkon belül bekövetkező hibákhoz saját hibajelzéseket rendelhetünk. Az **Error** adattípus implicit átalakítása nem lehetséges, vagyis a **CVErr( )** függvény eredménye közvetlenül egy nem **Variant** változónak nem adható át, ez csak típusának egy megfelelő függvénnyel (például **Cint( )**, **Cdbl( )** stb.) a kívánt adattípusra történő átalakítása után lehetséges.

Date

Szintaxis: **Date**

Ezt az argumentum nélküli függvény az operációs rendszer dátumát adja eredményül. A dátum beállítása a **Date** utasítással lehetséges.

DateAdd( )

Szintaxis: **DateAdd(** ***interval***, ***number***, ***date*** **)**

A függvény eredménye a névvel hivatkozott argumentumokban adott ***date*** dátumhoz a ***number*** értékének megfelelő számú ***interval*** hozzáadásával (kivonásával) kapott **Variant** adat, vagyis a függvénnyel egy adott dátumból/időpontból új dátumot/időpontot határozhatunk meg.

- A ***date*** argumentum egy tényleges dátum vagy egy, dátumot tartalmazó **Variant** neve is lehet.
- A ***number*** argumentumban pozitív vagy negatív számot, illetve ilyen eredményt adó numerikus kifejezést is használhatunk. A nem **Long** típusú ***number*** a **Fix( )** függvény szerint kerekítődik.
- Az adott dátumhoz adandó ***interval*** argumentum karakteres kifejezésében a következő szimbólumok valamelyikét használhatjuk:

interval	időegység:	***interval***	időegység:
yyyy	év	**w**	hétköznap
q	negyedév	**ww**	hét
m	hónap	**h**	óra
y	az év napja	**n**	perc
d	nap	**s**	másodperc

Ha a dátumhoz napokat kívánunk adni, az **y**, **d** vagy **w** jelű időegységek bármelyikét

használhatjuk Az új dátum számítása természetesen a hónapok hosszúságát és a szökőéveket is követi. A 100. évnél korábbi dátumot eredményező argumentumok hibát okoznak.

DateDiff( )

Szintaxis: **DateDiff(** ***interval***, ***date1***, ***date2*** [, ***firstdayofweek*** [, ***firstweekofyear***]])

A függvény a névvel hivatkozott argumentumok között megadott ***date1*** és ***date2*** dátumok különbségét az ***interval*** által meghatározott egységben, például negyedévben, napban vagy percben adja meg.

- A ***date1*** és ***date2*** megadása a **DateAdd( )** függvénynél már leírtakkal azonosan történhet. A függvényben a dátum kétféle, például "February 24, 1993" és #2/24/1993# alakja eltérő hatású lehet: az első alakot az évszám nélkül megadva, a dátumba mindig a rendszerben éppen aktuális évszám kerül. Ha a ***date2*** korábbi dátum mint a ***date1***, az eredmény negatív érték lesz.
- Az ***interval*** megadása ugyancsak a **DateAdd( )** függvénynél már leírtakkal azonos. Az ***interval*** argumentumként a két dátum közötti napok számításánál **y** vagy **d** időegységeket megadva egyaránt azonos eredményt kapunk. A hetek számolásánál a **w** (hétköznapi) és a **ww** (normál hetek) esetén a hetek száma az eltérő számolási módszer következtében különbözhet egymástól, ugyanis az ***interval*** **w** beállításnál az eredmény a ***date1*** és a ***date2*** között eltelt napok számában meglévő hétnapos egységek számát, a **ww** beállításnál pedig a naptári hetek, azaz a két dátum közötti vasárnapok számát kapjuk. A hetek számolásakor kapott eredményt a ***firstdayofweek*** értéke is befolyásolhatja.
- A függvény feltételes ***firstdayofweek*** argumentuma egy beépített állandó, amely a hét első napjaként használandó napot adja meg:

Ha a szimbolikus konstans:	Értéke:	A hét első napja:
vbUseSystem	0	Ha létezik, az alkalmazás beállítása, különben az NLS API beállítása szerinti
vbSunday	1	Vasárnap (alapértelmezett)
vbMonday	2	Hétfő
vbTuesday	3	Kedd
vbWednesday	4	Szerda
vbThursday	5	Csütörtök
vbFriday	6	Péntek
vbSaturday	7	Szombat

- Hasonlóképpen a ***firstweekofyear*** argumentum is az év első hetét adja meg:

Ha a szimbolikus konstans:	Értéke:	Az év első hete:
vbUseSystem	0	Ha létezik, az alkalmazás beállítása, különben az NLS API beállítása szerinti
vbFirstJan1	1	A január 1. tartalmazó hét (alapértelmezett)
vbFirstFourDays	2	Az évben legalább négy napot tartalmazó hét
vbFirstFullWeek	3	Az év első teljes hete

DatePart( )

Szintaxis: **DatePart(** ***interval***, ***date*** [, ***firstdayofweek*** [, ***firstweekofyear***]])

A függvény a ***date*** dátum egy adott részét adja eredményül. A névvel hivatkozott argumentumok (***interval***, ***date***, ***firstdayofweek*** és ***firstweekofyear***) értelmezésére és megadására a **DateDiff( )** függvénynél leírtak érvényesek.

A napokra vonatkozó eredményben az 1 vasárnapot, 2 hétfőt stb. jelöl. A hét meghatározásakor kapott eredményt a ***firstdayofweek*** értéke is befolyásolhatja.

DateSerial( )

Szintaxis: **DateSerial(** ***year***, ***month***, ***day***)

A függvény egy, a névvel hivatkozott ***year***, ***month*** és ***day*** argumentumokból képzett **Date** értéket ad eredményül.

A ***year*** argumentum 100 és 9999 közötti szám vagy ilyen számot eredményező numerikus kifejezés lehet, a ***month*** és ***day*** pedig bármely szám vagy numerikus kifejezés.

Kizárólag erre az évszázadra vonatkozóan az év helyén a teljes alak helyett 0 és 99 közötti számot is megadhatunk, így például a `DateSerial(5, 9, 11)` az 1905 szeptember 11 dátumnak felel meg. A függvényben argumentumként numerikus kifejezést használva az abszolút dátummegadás helyett egy rögzített dátumhoz viszonyított, relatív dátummegadás is lehetséges, például a `DateSerial(1993-9, 8-3, 1-1)` alak a May 31, 1984 dátumot adja vissza.

Ha a ***month*** vagy ***day*** argumentum értéke túllépi a természetes, ***month*** esetén 1 — 12, ***day*** esetén 1-31 közötti tartományát, az érték automatikusan a következő nagyobb időegységre konvertálódik. Bármelyik, az **Integer** számtartományból kilépő argumentum, valamint a megengedett dátumtartományból kilépő eredmény is hibát okoz.

DateValue( )

Szintaxis: **DateValue(** *dátum*)

A függvény a *dátum* argumentumban megadott bármely érvényes, 100. január 1. és 9999. december 31. közötti dátumként értelmezhető karakteres kifejezésnek vagy más, érvényes dátumot/időpontot tartalmazó kifejezésnek megfelelő **Date** értéket ad vissza.

Ha a *dátum* csak dátumtagoló karakterekkel elválasztott számokat tartalmaz, a függvény felismeri a rendszerben érvényes rövid dátumformának megfelelő karakterláncokat. Ugyancsak felismeri a rövid vagy teljes alakú hónapneveket tartalmazó, például November 15, 1996 vagy Aug 21, 1970 alakú dátumokat is. Ha a dátumból hiányzik az évszám, a függvény azt a rendszerben éppen érvényes évnek feltételezi.

Day()

Szintaxis: **Day(** *dátum* **)**

A függvény a *dátum* argumentum bármely érvényes, 100. január 1. és 9999. december 31. közötti dátumként értelmezhető numerikus, karakteres vagy ezek kombinációját tartalmazó kifejezéséből az év és hónap elhagyásával a napot emeli ki, és azt 1 és 31 közötti egész számként adja meg. Ha az argumentum **Null**, a függvény ugyancsak **Null** eredményt ad.

DDB()

Szintaxis: **DDB(** ***cost***, ***salvage***, ***life***, ***period*** [, ***factor***])

A gazdasági számításoknál használható **DDB( )** (*Double-Declining Balance*) függvény az argumentumban adott élettartamú vagyontárgy értékének egy adott periódusra vonatkozó értékcsökkenését, amortizációját adja meg az élettartam elején gyorsabb, majd mérsékeltebb értékvesztést feltételező degresszív módszer alapján.

- A függvény névvel hivatkozott argumentumai közül a ***cost*** a vagyontárgynak az élettartam elején érvényes értékét tartalmazza.
- A ***salvage*** a vagyontárgy maradványértékét, azaz az élettartam végén történő hasznosítással még visszanyerhető összeget adja meg.
- A ***life*** a vagyontárgy teljes használati ideje.
- A ***period*** az az időtartam, amelyre az értékcsökkenést meg kívánjuk határozni.
- A feltételes ***factor*** az alapértelmezett kettős értékcsökkenéstől eltérő mértékét írhat elő.

Az élettartamot és az értékcsökkenési időtartamot ugyanazon egységben (hónapban, évben) kell megadni. Értelemszerűen az argumentum minden elemének pozitív számnak kell lenni. A **DDB( )** függvény a ***period*** alatti értékcsökkenést a (***cost-salvage***)****factor/life*** összefüggés alapján számítja.

Dir()

Szintaxis: **Dir** [(*elérési_út* [, *attribútum*])]

A függvény az *elérési_út* argumentumban adott, fájlnévként értelmezhető karakteres

kifejezéshez illő nevű és az *attribútum* argumentum által előírt attribútumú fájl vagy könyvtár teljes, a meghajtónevet is tartalmazó elérési útjának meghatározására alkalmas. Az *elérési_út* a keresett fájl könyvtárát és meghajtóját is tartalmazhatja, továbbá a szokásos helyettesítő, un. wildcard (**?** és ***) karakterek használata is megengedett, így a többszörös fájlkijelölés is lehetséges.

A feltételes *attribútum* a következő táblázat szerinti Visual Basic állandó vagy az attribútumértékek összegét megadó numerikus kifejezés lehet:

Ha a szimbolikus konstans:	Értéke:	A fájl attribútuma:
vbNormal	0	*Normal* — normál (alapértelmezett)
vbHidden	2	*Hidden* — rejtett
vbSystem	4	*System* — rendszer
vbVolume	8	*Volume* — kötetcímke: érvényteleníti az egyéb attribútum-beállításokat
vbDirectory	16	*Directory* — könyvtár

Az *attribútum* értékei kombinálva is használhatók, így például a `Dir(*.*, 6)` az összes rejtett és rendszer fájlt is megadja. A kombinálás alól a lemezcímke jelent csak kivételt, amelynek használatakor a többi attribútumbeállítás figyelmen kívül marad. Hiányzó *attribútum* esetén a **Dir( )** függvény minden, az *elérési_út* által meghatározott, normál attribútumú fájt visszaad.

A **Dir( )** függvény első használatakor az *elérési_út* argumentumot kötelező megadni, egyébként hiba következik be, azonban az első fájl megtalálása után a következő illeszkedő fájl kereséshez az argumentumot már nem kötelező megadni. Az *elérési_út* az utolsó illeszkedő fájl megtalálása előtt is módosítható. A függvény sem a fájlokat, sem a könyvtárakat nem meghatározott sorrendben adja vissza, ezért a rendezésről külön kell gondoskodnunk.

Ha a függvény számára megadtuk *attribútum* argumentumot, kötelező az *elérési_út* megadása is, különben hiba következik be. Érvénytelen *elérési_út* esetén a függvény eredménye **Null**, sikertelen keresés esetén pedig üres ("") karakterlánc lesz. Sikertelen keresés után az *elérési_út* argumentumot ismét kötelező megadni.

A **Dir( )** függvény nem használható rekurzív módon.

DoEvents( )

Szintaxis: **DoEvents( )**

Az argumentum nélküli **DoEvents( )** függvény hatására a Visual Basic az operációs rendszer számára átadja a vezérlést, eredményként pedig a nyitott űrlapok számát adja vissza.

A függvény megszakítja a Visual Basic futását és a vezérlés mindaddig az operációs rendszernél marad, amíg az a várakozó teljes eseménysort fel nem dolgozza és a **SendKeys** billentyűműveleteit ki nem küldi, majd ezt követően a vezérlés visszaadódik a

DoEvents( ) utáni utasítás végrehajtására.

A függvény használata elsősorban olyan esetben hasznos, ha a Visual Basic program egyes részeinek végrehajtása olyan mértékben terhelheti a processzort, hogy az nem képes külső eseményekkel foglalkozni. Ennek veszélyét a **DoEvents( )** hívásoknak az adott programrészbe történő beillesztésével lehet megelőzni, így lehetővé téve a billentyűzetről jövő bemeneti adatok és egérkattintások hosszú késedelem nélküli fogadását és feldolgozását.

Az előre nem látható hatások elkerülésére ellenőrizni kell, hogy az az eljárás, amelyben a **DoEvents( )** függvény átadja a vezérlést, a program egy másik részében nem kerül ismét végrehajtásra, mielőtt az első **DoEvents( )** visszaadta volna a vezérlést. Ugyancsak kerülni kell a **DoEvents( )** függvény használatát, ha végrehajtása alatt más alkalmazások kapcsolatba léphetnek az azt tartalmazó eljárással.

Environ( )

Szintaxis: **Environ(** ***envstring*** | ***number*** **)**

Az **Environ( )** függvény az argumentumtól függően kétféle eredményt ad:

- A függvény első alakjában a névvel hivatkozott ***envstring*** argumentum egy, a környezeti változó nevét tartalmazó karakteres kifejezés, az eredmény pedig a változónévvel kijelölt, az operációs rendszer egy rendszerparaméterében beállított — vagyis a rendszerparaméterek karakterláncának táblázatában az egyenlőségjel után található — karakterlánc.
- A függvény második alakját használva a névvel hivatkozott ***number*** argumentumban egy számot vagy bármilyen érvényes numerikus kifejezést adhatunk meg, az eredmény pedig a rendszerazonosító táblázatának ennek megfelelő számú pozíciójában található, az ***envstring*** részt, azaz a környezeti változó nevét is tartalmazó teljes karakterlánc lesz. A ***number*** argumentumban adott numerikus kifejezés eredménye az **Int( )** függvény szerint kerekítődik. Ha a táblázatában az adott pozícióban nem található rendszerazonosítóra történő hivatkozás, az eredmény üres ("") karakterlánc lesz.

EOF( )

Szintaxis: **EOF(** *fájlszám* **)**

A függvény egy olyan logikai értéket ad vissza, amely jelzi, hogy már elértük-e az érvényes *fájlszám* argumentummal kijelölt fájl végét. Az érvényes *fájlszám* a fáj megnyitására használt **Open** utasításban szereplő számmal azonos, a függvény eredménye pedig az adott fájl végének elérése után **True** értékűre változik.

A véletlen vagy bináris elérésűként megnyitott fájlok esetén a függvény értéke **True**, ha az utolsóként végrehajtott **Get** utasítás nem egy teljes rekordot olvasott be.

Error()

Szintaxis: **Error** [(*hibaszám*)]

A függvény az argumentumban megadott számot hibaszámnak tekinti, és eredményül az ahhoz tartozó hibaüzenetet adja vissza.

A *hibaszám* bármely, a hibaszámokra megengedett tartományon (1 — 32767) belüli szám lehet, egyébként hiba következik be. Ha a megadott számhoz nem tartozik hibadefiníció, eredményként az "Application-defined or object-defined error" hibaüzenetet kapjuk vissza. A függvényt argumentum nélkül is használhatjuk, ekkor eredményként a legutolsóként bekövetkezett futásidejű hibának megfelelő hibaüzenetet, ennek hiányában pedig üres karakterláncot ("") kapunk. Ugyancsak üres karakterláncot ("") kapunk, ha a *hibaszám* nulla.

A legutolsóként bekövetkezett futásidejű hibát az **Err** objektum tulajdonságainak beállításait vizsgálva azonosíthatjuk. Az **Err** objektum **Description** tulajdonsága megegyezik az **Error( )** függvény eredményével.

Exp()

Szintaxis: **Exp(** *szám* **)**

A függvény a *szám* argumentumban adott bármely érvényes szám vagy numerikus kifejezés eredményét a természetes, vagyis e= 2,718282.. alapú hatvány kitevőjének tekintve meghatározza a hatványfüggvény értékét. Ha a kitevő értéke 709,782712893-nál nagyobb, hibajelzést kapunk.

FileAttr()

Szintaxis: **FileAttr(** ***filenumber***, ***returntype*** **)**

A függvény a névvel hivatkozott ***returntype*** argumentum értékétől függően a ***filenumber*** argumentumban megadott fájlszámmal az **Open** utasításban már megnyitott fájl elérési módját vagy az operációs rendszer fájlkezelési információit adja eredményül.

Ha a ***returntype*** argumentum értéke 1, az eredmény a következő táblázat szerint az adott fájl elérési módja lesz:

Ha az eredmény:	A fájlelérési mód:
1	**Input**
2	**Output**
4	**Random**
8	**Append**
32	**Binary**

Ha a ***returntype*** attribútum értéke 2, a függvény eredménye az operációs rendszer-

ben az adott fájlhoz tartozó fájlkezelő (fájlazonosító szám) lesz.

FileDateTime( )

Szintaxis: **FileDateTime(** *elérési_út* **)**

A függvény eredménye egy **VarType** 7 típusú dátumadat, amely az *elérési_út* argumentumban adott, fájlnévként értelmezhető karakteres kifejezéshez illő nevű fájl létrehozásának vagy legutóbbi módosításának dátumát és időpontját tartalmazza. Az *elérési_út* a keresett fájl könyvtárát és meghajtóját is tartalmazhatja.

FileLen( )

Szintaxis: **FileLen(** *elérési_út* **)**

A függvény az *elérési_út* argumentumban adott, fájlnévként értelmezhető karakteres kifejezésnek megfelelő nevű fájl byte-okban mért hosszát adja eredményül. Az *elérési_út* a keresett fájl könyvtárát és meghajtóját is tartalmazhatja.

A függvény meghívásakor már megnyitott fájl megadásakor az eredmény a fájlnak közvetlenül a megnyitása előtti hosszúságát jelöli. Nyitott fájl aktuális hosszúságát a **LOF( )** függvénnyel lehet meghatározni.

Fix( )

Szintaxis: **Fix(** *szám* **)**

A függvény a *szám* argumentumban adott szám vagy bármilyen érvényes numerikus kifejezés értékéből a tizedesek levágása utáni egészrészt adja meg, így például 13.6 esetén 13, míg -13.6 esetén -13 lesz az eredmény. A függvény a **Sgn(***szám***)*Int(Abs(***szám***))** kifejezéssel egyenértékű.

A függvény **Null** argumentum esetén **Null** eredményt ad.

Format( )

Szintaxis: **Format(** *kifejezés* [, ***format*** [, ***firstdayofweek*** [, ***firstweekofyear***]]])

A függvény a *kifejezés* argumentumban szereplő bármely érvényes kifejezést a ***format*** argumentumban szereplő, névvel hivatkozott vagy a felhasználó által definiált numerikus, dátum/idő vagy karakteres formázási kifejezés által meghatározott felépítésű karaktersorozattá alakítva ad vissza. Az előre definiált formázási alakok nevesítettek, a ***format*** argumentumban a nevüket kell megadni. Külön-külön formázási utasítások vonatkoznak a numerikus, a dátum/idő és a karakteres adatot tartalmazó *kifejezés* megjelenésének meghatározására.

Számok formázása

A számok vagy numerikus kifejezések esetén a Visual Basic által előre definiált vagy saját felépítésű formázást is használhatunk. Ha a *kifejezés* numerikus típusú, a ***format*** több, maximum négy saját felépítésű formázási utasítás szimbólumát tartalmazhatja, azokat egymástól pontosvesszővel elválasztva.

- Egyetlen formázási utasítás megadása esetén az vonatkozik minden értékre.
- Két formázási utasítás megadása esetén az első a nem negatív, a második pedig a negatív értékekre vonatkozik.
- Három formázási utasítás megadása esetén az első a pozitív, a második a negatív, a harmadik pedig a nullák formázására vonatkozik.
- Négy formázási utasítás megadása esetén az első a pozitív, a második a negatív, a harmadik a nullák, a negyedik pedig a **Null** értékek esetén lesz érvényes.

Az utóbbi két formázási alak esetén a két pontosvessző közötti valamelyik utasítást elhagyva, a hiányzó részre a pozitív számokra érvényes forma fog vonatkozni.

A numerikus kifejezések megjelenésének előre definiált, nevesített formázási lehetőségei a következők:

Név:	Hatása:
General Number	A számot folytonos alakban, az ezres helyiértékeket nem tagolva ábrázolja
Currency	A számot az ezres helyiértékeket tagolva, a negatív számokat pedig zárójelbe helyezve és a tizedesjel után két tizedesjegyet megadva ábrázolja. A numerikus kifejezés tényleges megjelenési formája a rendszer beállításától függ.
Fixed	A számot legalább egy egész és két tizedesjeggyel ábrázolja
Standard	A számot az ezres helyiértékeket tagolva és a tizedesjel után két tizedesjegyet megadva ábrázolja
Percent	A számot százzal megszorozva, két tizedesjegyet megadva és a számot a jobb oldalán % jellel kiegészítve ábrázolja
Scientific	A számot a szabványos tudományos alakban ábrázolja
Yes/No	Minden nem nulla **Yes**, a nulla pedig **No**
True/False	Minden nem nulla **True**, a nulla pedig **False**
On/Off	Minden nem nulla **On**, a nulla pedig **Off**

Ezen előre definiált formák esetén egyidőben csak egy formázási utasítást adható meg.

A függvény eredményének a felhasználó által definiált formázására a ***format*** kifejezésben a következő szimbólumok használhatók:

Szimbólum:	Hatása:
Üres karakter	Általános, nem formázott kijelzést eredményez.
0	Számjegyek helyfoglalója. Ha a kifejezés karakterláncában számjegy található a format karakterlánc 0 szimbólumának pozíciójában, az adott szám, egyébként pedig 0 jelenik meg. Ha a kifejezés a tizedesjegy előtt vagy után kevesebb jegyet tartalmaz, mint a format megadott 0 helyfoglalóinak száma, a hiányzó pozíciók 0 jegyekkel töltődnek ki. Ha a kifejezés több tizedesjegyet tartalmaz, csak az előírt számú, felfelé kerekített tizedesjegy kerül kijelzésre, de több egész esetén minden egészjegy kijelzésre kerül.
#	Számjegyek helyfoglalója. Ha a kifejezés karakterláncában számjegy található a forma karakterlánc # szimbólumának pozíciójában, az adott szám, egyébként pedig semmi sem jelenik meg. Ha a kifejezés a tizedesjegy előtt vagy után kevesebb jegyet tartalmaz, mint a format # helyfoglalóinak száma, a hiányzó pozíciók üresen maradnak. Ha a kifejezés több tizedesjegyet tartalmaz, csak az előírt számú, felfelé kerekített tizedesjegy, de több egész esetén minden egészjegy kijelzésre kerül.
.	Tizedesjel helyfoglalója, amely meghatározza, hogy hány egész- és tizedesjegy jelenjen meg a függvény eredményében. Ha a format a tizedesjel helyfoglalója előtt nem tartalmaz a számjegyek számára helyfoglalót, az egynél kisebb számok tizedesjellel kezdődnek. Ha ilyenkor a bevezető 0 számjegyre is szükség van, a tizedesjel helyfoglalója előtt a 0 (számjegy helyfoglaló) szimbólumot kell használni. A függvény eredményében tizedesjelként ténylegesen megjelenő karakter a rendszerben érvényes beállítástól függően "." vagy "," lesz.
%	Százalékjel helyfoglalója. A függvény eredményében a kifejezés számértékének százzal szorzott értéke jelenik meg, a százalékjel pedig a format karakterláncában lévő pozícióban kerül kijelzésre.
format	Ezres helyiértékek helyfoglalója. Az ezres helyfoglaló szimbólumát a format kifejezésben szokásosan a számjegyek helyfoglaló szimbólumai (0 vagy #) között kell használni. A tizedesjel mindkét oldalán 0 vagy # szimbólumot tartalmazó format kifejezés esetén a függvény eredményében a kifejezés ezres helyiértékű számjegyei pl. 3 451 459,5498. alakban tagolva jelennek meg. Speciális formázásra ad lehetőséget az ezres helyfoglalóknak a format kifejezésben közvetlenül a számjegyek és a tizedesjel helyfoglaló szimbólumai közötti (akár többszöri) elhelyezése, amelynek hatására a függvény eredményébe a kifejezés számértékének az elválasztójel számától függő számú ezerrel történő osztása és az eredmény kerekítése után kapott

értéke kerül, például a ###,,.00 formázási kifejezés a 43 541 612 számot 43.54, az 546 612 számot pedig .55 alakban adja vissza. Nem közvetlenül a tizedesjel helyfoglalója előtt használt helyfoglalók esetén az egyszeres és többszörös helyiérték helyfoglalók hatása azonos. A tagolásra ténylegesen használt karakter a rendszer beállításától függő lesz.

E- E+ e- e+ Tudományos formázási rendszer, amelyben a kifejezés számértéke a függvény eredményében az E vagy e szimbólummal elválasztott számra és kitevőre bontva jelenik meg. Ezen szimbólum után a kitevőben kívánt jegyek számának megfelelő, de legalább egy számjegy helyfoglaló szimbólumát (0 vagy #) kell elhelyezni. Az E+ és e+ szimbólumok használatakor a kitevőben mind a pozitív, mind a negatív, míg az E- és e- szimbólumok esetén csak a negatív előjel kerül kijelzésre.

- + $ ( ) szóköz Betűszerinti megjelenítés. Ha a függvény eredményében az itt szereplő szimbólumoktól eltérő karaktert kívánunk kiíratni, az adott karakter előtte egy " "\ karaktert kell elhelyezni, vagy a kiíratandó karaktert idézőjelek közé kell tenni.

**** A szimbólumot követő karaktert a függvény eredményének karakterláncába helyezi. Elsősorban olyan karakterek megjelenítéséhez használatos, amelyek a format kifejezésben szimbólumként, vagy a dátum- és időmegadásban jelentéssel bíró karakterként szerepelnek, így a számformázó karakterek (#, 0, %, E, e, vessző, pont), a dátum- és időpont-formázó karakterek (a, c, d, h, m, n, p, q, s, t, w, y, /) és a karakterlánc-formázó karakterek (@, &, <, >, !). A "\ "karakter a "\" szimbólummal együtt, vagyis "\\" alakban helyezhető az eredmény karakterláncába. A "\" szimbólum egyenértékű a megjelenítendő karakterek idézőjelek közé helyezésével.

"ABC" Az idézőjelek közé helyezett karakterláncot a függvény eredményének karakterláncába helyezi. A megjelenítendő, idézőjelek közötti karakterláncban szereplő idézőjelet annak ANSI kódjával, azaz Chr$(34)-gyel kell helyettesíteni.

Ha a ***format*** argumentum hiányzik vagy nulla hosszúságú (""), és a *kifejezés* egy numerikus kifejezés, a **Str( )** függvénnyel egyenértékű eredményt kapunk, azzal a jelentéktelen eltéréssel, hogy pozitív számok esetén az előjel számára nincs szóköz fenntartva.

A számkijelzés lehetséges formázási változataira a Visual Basic Súgó tartalmaz jellemző példákat.

Dátumok és időpontok formázása

Ha a *kifejezés* valós számot tartalmaz, azt dátum/időpont adatnak is tekinthetjük, és akár a Visual Basic előre definiált, akár saját formázási előírásainknak megfelelően jeleníthetjük meg. A dátum/időpont megjelenésének előre definiált, nevesített formázási lehetőségei a következők:

Név:	Hatása:
General Date	A dátumot és/vagy időpontot adja meg, a számból az egészeket dátumként, a tizedeseket időpontként értelmezve. A dátum/időpont tényleges megjelenési formája a rendszer beállításaitól függ.
Long Date	A rendszerben érvényes hosszú alakú dátumformát eredményez.
Medium Date	A Visual Basic területi beállításától függő közepes (betűvel kiírt hónapnevű) dátumformát eredményez.
Short Date	A rendszerben érvényes rövid alakú dátumformát eredményez.
Long Time	A rendszerben érvényes hosszú (óra, perc, másodperc) alakú időpontformát eredményez.
Medium Time	Az óra, perc felépítésű időpont a 12 órás rendszerben, a délelőtt/délután (AM/PM) jelzéssel kiegészítve jelenik meg.
Short Time	Az óra, perc felépítésű időpont 24 órás rendszerben jelenik meg.

A **Format** függvénnyel a dátumot/időpontot tartalmazó *kifejezés* karakterláncának felhasználó által definiált formázására a következő szimbólumok használhatók:

Szimbólum:	Hatása:
:	Időtagoló, vagyis az időpontot tartalmazó *kifejezés* megjelenítési formájában az óra, perc és másodperc értékei közötti elválasztójel. A **Format( )** függvény eredményében ténylegesen megjelenő karakter a rendszer beállításától függő lesz.
/	Dátumtagoló, vagyis a dátumot tartalmazó *kifejezés* megjelenítési formájában az év, hónap és nap értékei közötti elválasztójel. A **Format( )** függvény eredményében ténylegesen megjelenő karakter a rendszer beállításától függő lesz.
c	Ha van, a dátumot a **ddddd**, ezt követően pedig az időpontot a **ttttt** szimbólumnak megfelelően formázva adja meg.
d	A napokat 1, 2,..., 31 alakban adja meg.
dd	A napokat 01, 02,..., 31 alakban adja meg.
ddd	A napokat azok angol rövidítéseivel, (Mon, Thus,..., Sun) adja meg.
dddd	A napokat azok teljes angol nevével (Monday, Tuesday, ..., Sunday) adja meg.
ddddd	A *kifejezés*-ben szereplő teljes (év, hónap, nap) dátumot a rendszerben érvényes rövid dátumalaknak megfelelően formázva adja meg. (Az alapértelmezett forma: m/d/yy).

dddddd	A *kifejezés*-ben szereplő, teljes dátumként (év, hónap, nap) értelmezhető dátum-sorszámot a rendszerben érvényes hosszú dátumalaknak megfelelően formázva adja meg. (Az alapértelmezett forma: mmmm dd yyyy).
w	A hét napjait 1...7 közötti számként adja meg. (1=vasárnap, 2=hétfő, ...).
ww	Az év heteit 1...53 közötti számként adja meg.
m	A hónapokat 1, 2, ..., 12 alakban adja meg. Közvetlenül **h** vagy **hh** szimbólum után használva a hónapok helyett a perceket jelzi.
mm	A hónapokat 01, 02, ..., 12 alakban adja meg. Közvetlenül **h** vagy **hh** szimbólum után használva a hónapok helyett a perceket jelzi.
mmm	A hónapokat azok angol rövidítéseivel (Jan, Feb, ..., Dec) adja meg.
mmmm	A hónapokat azok teljes angol nevével (January, February, ..., December) adja meg.
q	Az év adott negyedévét 1 és 4 közötti számként adja meg.
y	Az év adott napját 1 és 366 közötti számként adja meg.
yy	Az évszámot annak utolsó két jegyével (például 93) adja meg.
yyyy	Az évszám teljes alakját (például 1993) adja meg.
h	Az órákat a 0, 1, 2, ..., 23 alakban adja meg.
hh	Az órákat a 00, 01, 02, ..., 23 alakban adja meg.
n	A perceket a 0, 1, 2, ..., 59 alakban adja meg.
nn	A perceket a 00, 01, 02, ..., 59 alakban adja meg.
s	A másodperceket 1, 2, ..., 59 alakban adja meg.
ss	A másodperceket 00, 01, ..., 59 alakban adja meg
ttttt	A *kifejezés* karakterláncot teljes alakú (óra, perc, másodperc) időpontként, a rendszerben érvényes időtagoló karakterekkel formázva adja meg. Az alapértelmezett forma: **h:mm:ss.**
AM/PM	Az időkijelzés a 12 órás rendszer szerinti lesz, éjféltől délig **AM**, déltől éjfélig **PM** kiegészítéssel.
am/pm	Az időkijelzés a 12 órás rendszer szerinti lesz, éjféltől délig **am**, déltől éjfélig **pm** kiegészítéssel.
A/P	Az időkijelzés a 12 órás rendszer szerinti lesz, éjféltől délig **A**, déltől éjfélig **P** kiegészítéssel.
a/p	Az időkijelzés a 12 órás rendszer szerinti lesz, éjféltől délig **a**, déltől éjfélig **p** kiegészítéssel.
AMPM, ampm	Az időkijelzés a 12 órás rendszer szerinti, 00:00 és 12:00 óra között a rendszerben a délelőtt jelzésére érvényes, 12:00 és 23:59 között pedig a délután jelzésére érvényes kiegészítő betűjelzéssel. Az alapértelmezett délelőtt/délután jelölés az AM/PM lesz.

A **Format( )** függvény névvel hivatkozott ***firstdayofweek*** feltételes argumentuma egy beépített szimbolikus konstans, amely a hét első napjaként használandó napot adja meg:

Ha a szimbolikus konstans:	Értéke:	A hét első napja:
vbUseSystem	0	Ha létezik, az alkalmazás beállítása, különben az NLS API beállítása szerinti
vbSunday	1	Vasárnap (alapértelmezett)
vbMonday	2	Hétfő
vbTuesday	3	Kedd
vbWednesday	4	Szerda
vbThursday	5	Csütörtök
vbFriday	6	Péntek
vbSaturday	7	Szombat

Hasonlóképpen a függvény névvel hivatkozott ***firstweekofyear*** argumentuma is egy beépített szimbolikus konstans, amely az év első hetét adja meg:

Ha a szimbolikus konstans:	Értéke:	Az év első hete:
vbUseSystem	0	Ha létezik, az alkalmazás beállítása, különben az NLS API beállítása szerinti
vbFirstJan1	1	A január 1. tartalmazó hét (alapértelmezett)
vbFirstFourDays	2	Az évben legalább négy napot tartalmazó hét
vbFirstFullWeek	3	Az év első teljes hete

A Visual Basic Súgó a különböző lehetséges dátum- és időpontformákra is számos hasznos példát mutat be.

Karakterláncok formázása

Végül, a **Format( )** függvényben a *kifejezés* argumentumot karakterláncként is kezelhetjük. Az ilyen argumentumhoz felhasználó által definiált, egy vagy két, egymástól pontosvesszővel elválasztott formázási utasítás szimbóluma tartozhat.

- Egy formázási utasítást megadása esetén minden karakterláncra ez fog vonatkozni.
- Két formázási utasítás megadása esetén az első a valódi karakterláncok, a második üres ("") karakterláncok és a **Null** értékek eset lesz érvényes.

A **Format( )** függvény eredményeként megjelenő karakterlánc formájának meghatározásához a következő formázási utasítások szimbólumait használhatjuk:

Szimbólum:	Hatása:
@	Karakter-helyfoglaló, amely a *kifejezés* karakterláncából az ugyanazon pozícióban található karaktert vagy egy szóközt jelenít meg. A függvény a helyfoglalókat általában jobbról balra tölti fel, kivéve, ha a ***format*** egy ! szimbólumot is tartalmaz.

$	Karakter-helyfoglaló, amely a *kifejezés* karakterláncából az ugyanazon pozícióban található karaktert vagy semmit sem jelenít meg. A függvény a helyfoglalókat általában jobbról balra tölti fel, kivéve, ha a ***format*** egy ! szimbólumot is tartalmaz.
<	A *kifejezés* karaktereit kisbetűs alakban adja vissza.
>	A *kifejezés* karaktereit nagybetűs alakban adja vissza.
!	A függvény a helyfoglalók pozícióit balról jobbra tölti fel a *kifejezés* karakterláncából.

FreeFile()

Szintaxis: **FreeFile** [(*számtartomány*)]

A **FreeFile( )** függvény eredménye az **Open** utasításban használható következő érvényes fájlszám. A *számtartomány* argumentum megszabja, hogy milyen tartományba eshet a függvény eredménye. Az alapértelmezett 0 érték esetén a visszaadott fájlszám 1 és 255 közötti, 1 érték esetén pedig 256 és 511 közötti lehet.

A függvény egy használni kívánt fájlszám érvényességének ellenőrzésére használatos.

FV()

Szintaxis: **FV(** ***rate***, ***nper***, ***pmt***[, ***pv*** [, ***type***]])

Ezzel az **FV( )** (*Future Value*) gazdasági függvénnyel egy adott futamidő végére az azonos részösszegű, rendszeres megtakarítás vagy kölcsöntörlesztés állandó kamat mellett elért jövőbeli értéke határozható meg a névvel hivatkozott argumentumoknak megfelelően.

- A ***rate*** argumentum a teljes futamidő alatt állandónak tekintett kamat, és mindig a pénzügyi műveletek (be- vagy kifizetések) közötti időközre kell vonatkoztatni (például az éves 12%-os kamat havonkénti időköznél 0.12/12=0.01).
- Az ***nper*** argumentum az összes pénzügyi műveletek száma (például 3 éves futamidő és havi gyakoriságú be- vagy kifizetéseknél 3*12=36).
- Az egyes pénzügyi műveletek során be- vagy kifizetett részösszeget a ***pmt*** argumentum tartalmazza, amely megtakarítás, bevétel esetén pozitív, kifizetés esetén negatív érték.
- A feltételes ***pv*** argumentum a jövőbeni összes ki- vagy befizetések jelenlegi értékét tartalmazza. Alapértelmezett értéke ebben a függvényben 0.
- Végül az ugyancsak feltételes ***type*** helyére 0 írandó, ha a pénzügyi műveletek a periódusok (például a tárgyhónapok) végén, míg 1, ha a periódusok elején történnek. Alapértelmezett értéke 0.

A függvényben a ***rate*** és az ***nper*** argumentumoknak azonos időközre, például hónapra vagy évre kell vonatkozniuk.

GetAllSettings( )

Szintaxis: **GetAllSettings(** ***appname, section*** **)**

A függvény a névvel hivatkozott ***appname*** argumentum karakteres kifejezésében adott nevű alkalmazás vagy projekt (pl. "MYAPP") Windows regisztrációs bejegyzésében (a 16-bites Windows esetén az .INI fájlban) található kulcsbeállítások listáját és azok megfelelő értékeit adja meg. A ***section*** argumentum karakteres kifejezése jelöli ki az adott alkalmazás azon résznek nevét (például "STARTUP"), amelynek beállításaira szükségünk van. A függvény eredménye a kulcsbeállításokat tartalmazó karakterláncokból álló kétdimenziós tömböt tartalmazó **Variant** érték. Ha bármelyik argumentum hiányzik, a függvény eredménye egy inicializálatlan **Variant** lesz.

GetAttr( )

Szintaxis: **GetAttr(** *elérési_út* **)**

A függvény az *elérési_út* argumentumban adott karakteres kifejezésben kijelölt fájl, könyvtár vagy kötet attribútumát jelöli. Az eredmény értelmezése a következő:

Ha a szimbolikus konstans:	Értéke:	A fájl attribútuma:
vbNormal	0	*Normal* — normál (alapértelmezett)
vbReadOnly	1	*Read-Only* — csak olvasható
vbHidden	2	*Hidden* — rejtett
vbSystem	4	*System* — rendszer
vbDirectory	16	*Directory* — könyvtár
vbArchive	32	*Archive* — az utolsó mentés óta módosított fájl

Az eredményből az attribútumok az **And** logikai operátor használatával, bitenkénti összehasonlítással állíthatók elő, például a MINTA nevű fájl csak olvasható attribútumú, ha a `GetAttr("MINTA") And vbReadOnly` kifejezés **True** eredményt ad.

GetObject( )

Szintaxis: **GetObject(** [***pathname***] [, ***class***] **)**

A függvény a feltételesen megadandó, névvel hivatkozott ***pathname*** argumentumban szereplő karakteres kifejezéssel megadott fájlból az OLE Automatizmushoz az ugyancsak feltételesen megadandó, névvel hivatkozott, ***class*** argumentumban megadott osztályú objektumot adja vissza. A keresett objektumot tartalmazó fájl megnevezésekor annak teljes, elérési útját is tartalmazó formáját kell használni a ***pathname*** argumentumban. Ha a ***pathname*** üres karakterlánc (""), a függvény az éppen aktív objektum egy

új példányát adja eredményül, Ha a ***pathname*** teljesen hiányzik, az egyébként feltételes ***class*** argumentumot kötelező megadni, és eredményül az adott osztály éppen aktív objektumát kapjuk vissza. Ha nincs a ***class*** osztálynak megfelelő objektum, a **GetObject( )** függvény hibajelzéssel tér vissza.

Az objektum osztályát előíró ***class*** karakteres kifejezés megadási alakja az *alkalmazásnév.objektumtípus*, ahol az *alkalmazásnév* az objektumot tartalmazó alkalmazást, az *objektumtípus* pedig a létrehozandó objektum típusát vagy osztályát jelöli.

A **GetObject( )** függvényt egy fájl OLE Automatizmus objektumához való hozzáférésre és a **Set** utasítással objektumváltozókénti beállítására lehet használni. Ha olyan alkalmazási programmal dolgozunk, amelynél egy fájl egy része is aktiválható, az ezen részt meghatározó ***pathname*** karakterlánc végére egy "**!**" szimbólumot kell helyezni. Az ilyen fájlok használatához elengedhetetlen az adott alkalmazás mélyebb ismerete.

Ha a függvény argumentumában nem jelöljük meg az objektum osztályát, az OLE határozza meg az elindítandó alkalmazást és az aktiválandó objektumot. Az olyan alkalmazás esetén, amely több objektumtípust is támogat, az aktiválandó objektumot kötelező kijelölni az egyébként feltételes ***class*** argumentum megadásával.

A **GetObject( )** függvényt akkor lehet használni, ha az objektumnak már létezik példánya, vagy azt egy már betöltött fájlal kívánjunk létrehozni. Ha még nincs objektumpéldány és nem akarjuk az objektumot egy fájl betöltésével elindítani, a **CreateObject( )** függvényt kell használni. A magát egyetlen példányban deklaráló objektumra a **GetObject( )** függvényt akárhányszor is hajtjuk végre üres karakterláncot ("") tartalmazó ***pathname*** argumentummal, mindig ugyanazon objektumpéldányt kapjuk vissza. Az ilyen, magát egyetlen példányban deklaráló objektum esetén a ***pathname*** elhagyása hibát okoz.

A **GetObject( )** függvény nem használható a Visual Basic által létrehozott osztályra történő hivatkozáshoz.

GetSetting( )

Szintaxis: **GetSetting(** ***appname, section, key*** [, ***default***] **)**

A függvény egy, a névvel hivatkozott ***appname*** argumentum karakteres kifejezésében adott nevű alkalmazás vagy projekt (pl. "MYAPP") Windows regisztrációs bejegyzésében (a 16-bites Windows esetén az .INI fájlban) található beállítás értékét adja meg. A ***section*** argumentum karakteres kifejezése tartalmazza az adott alkalmazás azon résznek nevét (például "STARTUP"), amelyben a keresett beállítás található. A beállítás nevét (például "LEFT"), a ***key*** argumentum karakteres kifejezéseként kell megadni. Ha a függvény nem talál beállítási értéket, eredményül a feltételes ***default*** argumentum karakteres kifejezésében szereplő értéket adja vissza. A ***default*** alapértelmezetten üres ("") karakterlánc.

Ha a függvény bármelyik kötelező argumentuma hiányzik, az eredmény a ***default*** tartalma lesz.

Hex()

Szintaxis: **Hex(** *szám* **)**

A függvény a *szám* argumentumban lévő bármely érvényes numerikus vagy karakteres kifejezés legközelebbi egészre kerekített értékének hexadecimális számrendszerbeli alakját legfeljebb nyolc karakterből álló karakterláncként adja meg. Ha a *szám* argumentum **Null**, a függvény eredménye is **Null**, ha pedig **Empty**, az eredmény 0 lesz.

A kifejezésekben közvetlenül használandó oktális értékeket az **&H** előtaggal kell megkülönböztetni az egyéb karakterláncoktól.

Hour()

Szintaxis: **Hour(** *idő* **)**

A függvény az *idő* argumentumban lévő bármely érvényes, időpontként értelmezhető numerikus vagy karakteres kifejezésből, vagy ezek kombinációjából csak az órát jelző részt emeli ki, és eredményül egy annak megfelelő, 0 és 23 közötti egész számot ad meg. Ha a *szám* argumentum **Null**, a függvény eredménye is **Null** lesz.

IIf()

Szintaxis: **IIf(** ***expr***, ***truepart***, ***falsepart*** **)**

A függvény eredménye a névvel hivatkozott ***expr*** argumentumban lévő tetszőleges kifejezés logikai értékétől függ. Ha az ***expr*** kifejezés kiértékelésének eredménye **True**, az eredmény a ***truepart***, ha pedig **False**, a ***falsepart*** argumentumokban lévő érték vagy kifejezés lesz.

A függvény használatakor figyelemmel kell lenni arra, hogy a névvel hivatkozott ***truepart*** és ***falsepart*** argumentumokban adott kifejezések mindegyike — az ***expr*** kifejezés logikai értékétől függetlenül — kiértékelésre kerül, így például a ***falsepart*** argumentumban szereplő kifejezésben egy nullával való osztás hatására az **IIf( )** függvény akkor is hibát ad vissza, ha az ***expr*** kifejezés kiértékelésének eredménye **True** volt.

IMEStatus()

Szintaxis:

A csak távol-keleti változatokban használatos, argumentum nélküli függvény az éppen érvényes **Input Method Editor** (IME) módot adja vissza. Az eredmény értelmezését lásd a Visual Basic Súgóban.

Input( )

Szintaxis: **Input(** *szám*, [#] *fájlszám*)

A függvény a *fájlszám* argumentumban adott bármely érvényes fájlszámú, **Input** vagy **Binary** módban már megnyitott fájlból a *szám* argumentumban adott darab karaktert olvas be.

Eltérően az **Input#** utasítástól, az **Input( )** függvény minden karaktert, így a kezdő szóközt, idézőjelet, vesszőt, kocsivissza és soremelés karaktereket is visszaadja. A *fájlszám* a fáj megnyitására használt **Open** utasításban szereplő számmal azonos.

InputB( )

Szintaxis: **InputB(** *szám*, [#] *fájlszám*)

Az **InputB( )** az **Input( )** függvényhez hasonló funkciójú, azonban a *fájlszám* argumentumban adott, bármely érvényes fájlszámú, **Input** vagy **Binary** módban már megnyitott szövegfájlból **Byte** típusú adatok beolvasására használható. A beolvasandó byteok számát a *szám* argumentumban kell megadni.

InputBox( )

Szintaxis: **InputBox(*prompt*** [, ***title***] [, ***default***] [, ***xpos***] [, ***ypos***] [, ***helpfile***, ***context***])

Az **InputBox( )** függvény az alkalmazás futásakor a képernyőn a felhasználót tájékoztató üzenetet tartalmazó párbeszédpanelt hoz létre, várakozik a felhasználó szöveg beírásával vagy egy nyomógomb lenyomásával történő válaszadására, majd a szövegbeviteli mező tartalmát adja vissza.

- A párbeszédpanelen megjelenő üzenet karakterláncát a függvény névvel hivatkozott ***prompt*** argumentumában kell megadni. Az üzenet a használt karaktertípus szélességétől függően kb. 1024 karakternyi hosszú lehet. A többsoros üzeneteket az egyes sorok közé a kocsivisssza, azaz **Chr**(13) és a soremelés, azaz **Chr**(10) karakterek, illetve ezek kombinációjának **Chr**(13) **&** **Chr**(10) alakban történő elhelyezésével lehet tördelni.

A párbeszédpanel megjelenését néhány feltételes, névvel hivatkozott argumentummal lehet befolyásolni.

- A ***title*** argumentumban a párbeszédpanel címsorában megjeleníteni kívánt szöveget tartalmazó karakteres kifejezést adhatjuk meg, ennek elhagyásakor a címsor az alkalmazás nevét tartalmazza.
- A párbeszédpanel szövegbeviteli mezőjében alapértelmezett felhasználói válaszként megjeleníteni kívánt szöveget a ***default*** argumentum karakteres kifejezésé-

ben helyezhetünk el. Az argumentum elhagyása üres szövegbeviteli mezőt eredményez.

- A párbeszédpanel képernyőn belüli pozícióját az ***xpos*** és az ***ypos*** argumentumokban adott numerikus kifejezésekben a párbeszédpanel bal oldali, illetve felső élének a képernyő bal, illetve felső szélétől mért távolságát megadva írhatjuk elő. A numerikus kifejezések értéke úgynevezett *twip* egységként értelmezett: a twip egy pont huszad része, így 1 (logikai) cm = 567 twip. Ha az ***xpos*** argumentum hiányzik, a párbeszédpanel a képernyőn vízszintes irányban középre igazítva, míg az ***ypos*** argumentum hiányában a képernyő felső élétől kb. egyharmad képernyőmagasságra igazítva jelenik meg.
- A ***helpfile*** és a ***context*** argumentumok használatával súgót rendelhetünk a függvénnyel megjelenítendő párbeszédpanelhez. Ilyenkor a párbeszédpanel automatikusan tartalmaz egy Help nyomógombot is. A ***helpfile*** karakteres kifejezés, amelyben az adott párbeszédpanelhez tartozó magyarázószöveget tároló fájl nevét kell megadni, a ***context*** pedig numerikus kifejezés, amely a súgófájlon belüli téma számát jelöli ki. A ***helpfile*** és a ***context*** argumentumokat csak együtt lehet megadni vagy elhagyni.

A függvény a párbeszédpanelen az OK választása, illetve az ENTER billentyű leütése után a szövegbeviteli mező tartalmát (az alapértelmezett vagy a felhasználó által beírt választ), a Cancel választása után pedig a szövegbeviteli mező tartalmától függetlenül üres ("") karakterláncot ad eredményül.

Az argumentumlista bármelyik feltételes tagja elhagyható, azonban az előtte lévő elválasztó vesszőt ki kell tenni.

InStr()

Szintaxis: **InStr**([***start***], ***string1***, ***string2*** [, ***compare***])

A függvény eredménye a névvel hivatkozott ***string1*** argumentum karakteres kifejezésében a ***string2*** argumentumban lévő karakteres kifejezés első előfordulásának helyzete.

- A ***start*** argumentum egy numerikus kifejezés, amely a ***string1*** kifejezésben történő a keresés kezdő karakterpozícióját határozza meg. A ***start*** alapértelmezett értéke 0. Noha a ***start*** argumentum feltételes, a ***compare*** használata esetén kötelező megadni. A **Null** értékű ***start*** megadása hibát eredményez.
- A feltételes ***compare*** argumentum a ***string1*** és a ***string2*** összehasonlítás típusát írja elő. A ***compare*** használata esetén kötelező megadni a ***start*** argumentumot is. Az összehasonlítási típusok a következők lehetnek:
  - Ha a ***compare*** hiányzik, az összehasonlítás az **Option Compare** utasítással előírt módon történik.

- Ha a ***compare*** = 0 (alapértelmezett érték), az összehasonlítás binárisan történik.
- Ha a ***compare*** = 1, az összehasonlítás szövegek összehasonlításaként, a nagy- és kisbetűs karaktereket azonosnak tekintve történik.
- Ha a ***compare*** = **Null**, hiba generálódik.
- Ha a ***compare*** = egy **Field** objektum **CollatingOrder** tulajdonságának értéke, egy adatbázis értékeinek rendezése vagy összehasonlítása történik, ugyanolyan módon, ahogy azt az adatbázis végezné.

Szokásos esetben az **InStr( )** függvény eredménye egy szám, amely a sikeres keresésnél a ***string1*** karakteres kifejezés ***start*** induló pozíciója és a ***string2*** első eleme közötti pozíciók száma. A sikertelen keresés, üres karakterláncot ("") tartalmazó ***string1*** vagy a ***start*** > ***string1*** esetben az eredmény 0 lesz. Ha a ***string2*** üres karakterlánc (""), az eredmény a ***start*** számértékét kapja. Amennyiben akár a ***string1***, akár a ***string2*** **Null**, a függvény eredménye is **Null** lesz.

InStrB( )

Szintaxis: **InStrB(** [***start***], ***string1***, ***string2*** [, ***compare***])

A **Byte** típusú adatokat tartalmazó karakterlánc esetén használható **InStrB( )** az **InStr( )** függvényhez hasonló funkciójú, azonban az eredmény a ***string1*** karakteres kifejezésében a ***string2*** első előfordulásának karakterpozíciója helyett a byte-pozíciót adja meg.

Int( )

Szintaxis: **Int(** *szám*)

A függvény egy szám vagy bármilyen érvényes numerikus kifejezés egész részét adja meg. Pozitív *szám* esetében az eredmény megegyezik a **Fix( )** függvény eredményével, azonban a negatív *szám* esetén az ahhoz legközelebbi, de nála kisebb negatív egész számot kapjuk meg, így például 13.6 esetén 13, míg -13.6 esetén -14 lesz az eredmény.

A függvény **Null** argumentum esetén **Null** eredményt ad.

IPmt( )

Szintaxis: **IPmt(** ***rate***, ***per***, ***nper***, ***pv*** [, ***fv*** [***type***]])

Ez az **IPmt( )** (*Interest Payment*) olyan gazdasági függvény, amelyet adott futamidő, valamint állandó részletfizetések és kamatfeltételek mellett egy végösszegre (kölcsöntörlesztés vagy megtakarítás) vonatkozóan adott időszakra érvényes kamatösszeget adja meg.

- A névvel hivatkozott argumentumok közül a ***rate***, az ***nper***, a ***pv*** és a ***type*** értelmezése azonos az **FV** függvénynél leírtakkal.
- A ***per*** egy, értelemszerűen csak 1 és a ***nper*** közötti szám, és azt adja meg, hogy hány pénzügyi műveletre (ki- vagy befizetésre) vonatkozóan kívánjuk a kamatösszegzést elvégezni.
- Az ***fv*** helyére a futamidő végére elérni kívánt megtakarítás összegét, törlesztések esetében értelemszerűen nullát kell beírni. Alapértelmezett értéke 0.

IRR( )

Szintaxis: **IRR(** *values*( )[, ***guess***])

Ezzel az **IRR( )** (*Internal Rate of Return*) pénzügyi függvénnyel egy rendszeres pénzforgalmat lebonyolító számlán adott időszak alatt elért relatív kamat mértékét lehet meghatározni.

- A névvel hivatkozott ***values*( )** argumentum egy adattömb, amelynek legalább egy negatív értéket (kifizetést) és egy pozitív értéket (bevétel) kell tartalmaznia. A tömb elemei csak számértékek lehetnek, numerikus kifejezést nem használhatunk. Mivel az eredmény természetszerűleg az adattömb elemeinek sorrendjétől függ, erre a tömb megadásakor figyelemmel kell lenni. A tömb elemei eltérő értékeket is tartalmazhatnak.
- A feltételes ***guess*** argumentumban egy becsült kamatszintet lehet megadni, amelyből a függvény legfeljebb húsz iterációs lépéssel kísérli meghatározni a tényleges értéket. Ha húsz iterációs lépés után sem jobb az eredmény pontossága mint 0.00001%, a számítás sikertelen lesz. A ***guess*** alapértelmezetten 0.1, azaz 10%. Ha a függvény az iterációs eljárás végén nem talál megfelelő végeredményt, más becsléssel a függvényt újra kell indítani.

IsArray( )

Szintaxis: **IsArray(** *változónév*)

A függvény eredménye **True**, ha a *változónév* argumentumként megadott bármilyen változó tömb, egyébként **False**.

Az **IsArray( )** elsősorban tömböket tartalmazó **Variant** értékek vizsgálatára alkalmazható.

IsDate()

Szintaxis: **IsDate(** *kifejezés* **)**

A függvény eredménye **True**, ha a *kifejezés* argumentumként megadott bármilyen dátum vagy dátumként/időpontként értelmezhető érvényes karakteres kifejezés (a Microsoft Windowsban) 100. január 1. és 9999. december 31. közötti dátumot jelöl, egyébként **False**. Az érvényes dátumhatárok az operációs rendszertől függenek.

IsEmpty()

Szintaxis: **IsEmpty(** *kifejezés* **)**

A függvény eredménye **True**, ha a *kifejezés* argumentumként megadott bármilyen numerikus vagy karakteres kifejezésben lévő egyedi változó még nem inicializált vagy explicit módon **Empty** értékűre állított. Az inicializált változó vagy a *kifejezés* argumentumban egynél több változó **False** eredményt ad. Az **Empty** értéktől különbözik a **Null**, amely üres adatot vagy adatot nem tartalmazó, de már inicializált változót jelez.

IsError()

Szintaxis: **IsError(** *kifejezés* **)**

A függvény eredménye **True**, ha a *kifejezés* argumentumként megadott **VarType vbError** típusú **Variant** *kifejezés* hibaérték, egyébként **False**. Hibaértékek létrehozásához a valós számokat a **CVErr( )** függvénnyel lehet hibaértékké alakítani.

Az **IsError( )** függvénnyel egy numerikus kifejezésről lehet megállapítani, hogy az hibát jelöl-e.

IsMissing()

Szintaxis: **IsMissing(** *argumentumnév* **)**

A függvény eredménye **True**, ha az *argumentumnév* kifejezésben megadott feltételes eljárásargumentum átadása egy eljárás számára nem történt meg, egyébként **False**.

Az **IsMissing( )** függvényt **ParamArray** argumentumon használva mindig **False** eredményt kapunk. Ha egy argumentum esetén az eredmény **True**, a hiányzó *argumentumnév* másik kódban történő használata felhasználó által definiált hibát okoz.

IsNull()

Szintaxis: **IsNull(** *kifejezés* **)**

A függvény eredménye **True**, ha a *kifejezés* argumentumként megadott bármilyen

numerikus vagy karakteres kifejezésben **Null** érték van, egyébként **False**. A **Null** érték azt jelzi, hogy a **Variant** nem tartalmaz érvényes adatot, de ez nem azonos az üres ("") karakteres kifejezéssel, vagy az **Empty** értékkel, amely csak a változó inicializálatlanságát jelöli.

A több változót tartalmazó *kifejezés* bármely változójának **Null** értéke a teljes *kifejezés* **Null** értékét eredményezi, így például az `If Var = Null` vagy az `If Var > Null` kifejezés a kiértékelés eredményétől függetlenül is **Null** lesz.

IsNumeric( )

Szintaxis: **IsNumeric(** *kifejezés* **)**

A függvény eredménye **True**, ha a *kifejezés* argumentumként megadott bármilyen numerikus vagy karakteres kifejezés numerikus adattípusként értékelhető ki, egyébként **False**.

Ha a *kifejezés* dátumkifejezést tartalmaz, az eredmény **False** lesz.

IsObject( )

Szintaxis: **IsObject(** *kifejezés* **)**

A függvény eredménye **True**, ha a *kifejezés* argumentumként megadott bármilyen kifejezés egy érvényes OLE Automatizmus objektumra hivatkozik (azaz egy **Object** típus változója, **VarType vbObject** típusú **Variant** vagy felhasználó által definiált objektum), egyébként **False**.

LBound( )

Szintaxis: **LBound(** *tömbnév* [, *dimenzió*] **)**

A függvény a *tömbnév* argumentummal kijelölt nevű adattömbben a *dimenzió* argumentumban adott számnak megfelelő dimenziója számára definiált legkisebb indexet adja vissza. A *dimenzió* alapértelmezetten 1, így megadása egydimenziós tömböknél elhagyható.

Az olyan, például `MyArray(21)` alakú definíció esetén, amely nem explicit módon dimenzionálja a tömböt, a legkisebb index az **Option Base** értékétől függően csak 0 vagy 1 lehet. A **Dim**, **Private**, **Public**, **ReDim** vagy **Static** utasításokban a **To** kiegészítő alakkal dimenzionált tömbök mindig közvetlenül írják elő a legkisebb index értéket, és az tetszőleges egész szám lehet.

Az **Lbound( )** és **Ubound( )** függvények együttesen egy tömb méretének meghatározására használhatók.

LCase( )

Szintaxis: **LCase(** *karakterlánc*)

A függvény a *karakterlánc* argumentumban adott bármely érvényes karakteres kifejezést kisbetűs alakban — a nagybetűs karaktereket kisbetűs alakkal helyettesítve, az összes többi karaktert változatlanul hagyva — adja vissza. Ha a *karakterlánc* karakteres kifejezése **Null**, az eredmény is **Null** lesz.

Left( )

Szintaxis: **Left(** ***string***, ***length***)

A függvény eredménye a névvel hivatkozott ***string*** argumentumban adott karakteres kifejezés bal oldaláról a ***length*** argumentumban szereplő numerikus kifejezéssel megadott számú karakter lesz.

- Ha a ***string*** karakteres kifejezése **Null**, az eredmény is **Null**.
- Ha a ***length*** értéke 0, az eredmény üres ("") karakterlánc, míg ha a ***length*** nagyobb, mint a karakteres kifejezés hosszúsága, az eredmény a teljes kifejezés lesz.

A karakteres kifejezés hosszát a **Left( )** használata előtt a **Len( )** függvénnyel lehet kiolvasni.

LeftB( )

Szintaxis: **LeftB(** ***string***, ***length***)

A **Byte** típusú adatokat tartalmazó karakterláncok esetén használható **LeftB( )** a **Left( )** függvényhez hasonló funkciójú, azonban a ***length*** argumentum nem a visszaadandó karakterek, hanem a byte-ok számát határozza meg.

Len( )

Szintaxis: **Len(** *karakterlánc* | *változónév*)

A függvény vagy a *karakterlánc* argumentumban lévő bármely érvényes karakteres kifejezés karaktereinek számát adja meg.

Ha **Len( )** függvény *változónév* argumentumot tartalmazó alakját használjuk, eredményül a megadott érvényes változó tárolásához szükséges byte-ok számát kapjuk visszsza. Ez alól kivétel a **Variant** értéket tartalmazó *változónév* argumentum, amelyet a függvény a **String** típussal azonos módon kezel, vagyis az argumentum karaktereinek számát adja vissza. A függvény a felhasználó által definiált adattípust tartalmazó *válto-*

zónév argumentumra nem használható.

Ha a **Len( )** függvény bármelyik (*karakterlánc* vagy *változónév*) argumentuma **Null**, az eredmény ugyancsak **Null** lesz.

LenB( )

Szintaxis: **LenB(** *karakterlánc* | *változónév*)

A **Byte** típusú adatokat tartalmazó karakterláncok esetén használható **LenB( )** a **Len( )** függvényhez hasonló funkciójú, azonban a *karakterlánc* argumentum karaktereinek száma helyett a byte-ok számát adja meg.

LoadPicture( )

Szintaxis: **LoadPicture(** [*karakteres_kifejezés*])

A függvény a *karakteres_kifejezés* argumentumban megadott nevű grafikus fájl tartalmát egy **Form** objektumba, **PictureBox** vezérlőelembe vagy **Image** vezérlőelembe tölti. A Visual Basic a következő grafikus fájlokat ismeri: bitkép (.BMP), ikon (.ICO), futáshossz-kódolású fájlok (.RLE) és metafájl (.WMF).

A grafikának megjelenítéshez a **LoadPicture( )** függvény eredményét egy **PictureBox** vagy **Image** vezérlőelem, illetve **Form** objektum **Picture** tulajdonságának, ikonok megjelenítéséhez pedig egy **Form** objektum **Icon** tulajdonságának kell adni. Ikon betöltése a **LoadPicture( )** függvény eredményének a **Timer** és **Menu** kivételével mindegyik vezérlőelem **DragIcon** tulajdonságának adva is lehetséges.

A függvényt a feltételes argumentum nélküli alakban használva az objektumokból vagy vezérlőelemekről a grafikák törölhetők.

LoadResData( )

Szintaxis: **LoadResData(** *index, forma*)

A függvény egy erőforrásfájlból (.RES) az *index* argumentumban adott integer értékkel vagy karakterlánccal adott adatazonosítónak (ID) megfelelő adatot tölti be és eredményül egy **Byte** adatot ad vissza. Az ID = 1 adatazonosító az alkalmazás ikonja számára van fenntartva. A *forma* argumentum értéke az adat eredeti formáját határozza meg a következő táblázatnak megfelelően:

Ha a *forma*:	Az eredeti erőforrás:
1	Mutató
2	Bitkép
3	Ikon
4	Menü

5	Párbeszédpanel
6	Karakterlánc
7	Fontkönyvtár
8	Font
9	Gyorsítótábla
10	Felhasználó által definiált
12	Csoport mutató
14	Csoport ikon

A függvény egy bitképpel, ikonnal vagy mutató erőforrástípussal az erőforrás aktuális bitjeit tartalmazó karakterláncot ad vissza. A függvény által az erőforrásfájlból betölthető adatmennyiség maximálisan 64K lehet. Ha az aktuális bitképet, ikont vagy erőforrást használni kívánjuk, a **LoadResPicture( )** függvényt kell használnunk.

A **LoadResData( )** függvény különösen a Visual Basic alkalmazások nemzeti változatainak elkészítés során hasznos, mivel így a fordítandó erőforrások egy erőforrásfájlban vannak elkülönítve, és ezért szükségtelen a programkód módosítása vagy az alkalmazás újrafordítása.

LoadResPicture()

Szintaxis: **LoadResPicture(** *index, forma* **)**

A függvény egy erőforrásfájlból (.RES) az *index* argumentumban adott integer értékkel vagy karakterlánccal adott adatazonosítónak (ID) megfelelő bitképet, ikont vagy mutatót tölti be. Az ID = 1 adatazonosító az alkalmazás ikonja számára van fenntartva. A *forma* argumentum értéke a visszaadott adat formáját határozza meg a következő táblázatnak megfelelően:

Ha a szimbolikus konstans:	Értéke:	Az eredeti erőforrás:
vbResBitmap	0	Bitkép
vbResIcon	1	Ikon
vbResCursor	2	Mutató

A **LoadResPicture( )** függvény használatával egy **Form** objektum vagy vezérlőelem **Picture** tulajdonságára való hivatkozás nélkül lehet grafikákat betölteni. A bitképek, ikonok és mutatók erőforrásfájlban történő tárolása és onnan történő használata lényegesen megrövidíti a betöltésükhöz szükséges időt, mivel az szükség szerint, egyedileg történhet, ellentétben az összes grafikának egyszerre, a **Form** objektummal együtt történő betöltésével.

A **LoadResPicture( )** függvény különösen a Visual Basic alkalmazások nemzeti változatainak elkészítés során hasznos, mivel így a fordítandó erőforrások egy erőforrásfájlban vannak elkülönítve, és ezért szükségtelen a programkód módosítása vagy az alkalmazás újrafordítása.

LoadResString( )

Szintaxis: **LoadResString(** *index* **)**

A függvény egy erőforrásfájlból (.RES) az *index* argumentumban adott integer értékkel vagy karakterlánccal adott adatazonosítónak (ID) megfelelő karakterláncot tölti be. Az ID = 1 adatazonosító az alkalmazás ikonja számára van fenntartva.

A **LoadResString( )** függvény használatával elkerülhető a programkódban a betűszerinti karakterláncok használata. Az adatok hosszú karakterláncainak erőforrásfájlban történő tárolása és onnan történő használata lényegesen megrövidíti a betöltésükhöz szükséges időt, mivel az szükség szerint, egyedileg történhet, ellentétben az összes grafikának egyszerre, a **Form** objektummal együtt történő betöltésével.

A **LoadResString( )** függvény különösen a Visual Basic alkalmazások nemzeti változatainak elkészítés során hasznos, mivel így a fordítandó erőforrások egy erőforrásfájlban vannak elkülönítve, és ezért szükségtelen a programkód módosítása vagy az alkalmazás újrafordítása.

Loc( )

Szintaxis: **Loc(** *fájlszám* **)**

A függvény *fájlszám* argumentumaként az **Open** utasításban a fájl megnyitására használt számot használva, az eredmény a már megnyitott fájlban aktuális írási/olvasási pozíciót adja vissza: a **Random** elérésű fájloknál az utolsóként olvasott vagy írt rekord számát, a **Sequential** elérésű fájloknál az éppen érvényes byte-pozíció 128-ad részét, **Binary** fájloknál pedig az utolsóként olvasott vagy írt byte pozícióját adja.

LOF( )

Szintaxis: **LOF(** *fájlszám* **)**

A függvény *fájlszám* argumentumaként az **Open** utasításban a fájl megnyitására használt számot használva, az eredmény az adott, már megnyitott fájl byte-okban mért hossza lesz. Nem megnyitott fájlnál a fájl hossza a **FileLen( )** függvénnyel olvasható ki.

Log( )

Szintaxis: **Log(** *szám* **)**

A függvény a *szám* nevű argumentumban adott bármely érvényes, nullánál nagyobb eredményt adó numerikus kifejezés értékének természetes, vagyis e = 2,718282... alapú logaritmusát adja vissza. A kapott eredményből a tetszőleges *n* alapú logaritmus meghatározása a **Log(***n***)** értékkel történő osztással lehetséges.

LTrim( )

Szintaxis: **LTrim(** *karakterlánc* **)**

A függvény eredménye a *karakterlánc* argumentumban szereplő bármely érvényes karakteres_kifejezés bal oldali (azaz bevezető) szóközök nélküli másolata. Ha az argumentum **Null** típusú, az eredmény is **Null** lesz.

Mid( )

Szintaxis: **Mid(** ***string***, ***start*** [, ***length***] **)**

A függvény a névvel hivatkozott ***string*** argumentumban adott bármilyen karakteres kifejezésből egy, a ***start*** által meghatározott számú karakterpozíciótól kezdődő, és a ***length*** argumentumban adott számú karakterből álló karakterláncot ad vissza.

- Ha a karakteres kifejezés **Null** típusú, az eredmény is **Null** lesz.
- Ha a ***length*** feltételes argumentum hiányzik, vagy az általa meghatározott hosszúság nagyobb, mint a ***start*** pozíció utáni karakterek száma, az eredmény a ***string*** kifejezés ***start*** pozíció utáni része lesz.
- Ha a függvényben a ***start*** pozíció nagyobb mint a ***string*** karakteres kifejezés karaktereinek száma, szintén üres ("") karakterláncot kapunk eredményül.

A ***string*** karakteres kifejezésben található karaktereinek számát a **Len( )** függvénnyel lehet meghatározni.

MidB( )

Szintaxis: **MidB(** ***string***, ***start*** [, ***length***] **)**

A **Byte** típusú adatokat tartalmazó karakterláncok esetén használható **MidB( )** a **Mid( )** függvényhez hasonló funkciójú, azonban a ***string*** argumentumból visszaadandó részt a ***start*** és ***length*** argumentumok a karakterek száma helyett a byte-ok számával adják meg.

Minute( )

Szintaxis: **Minute(** *idő* **)**

A függvény az *idő* argumentumban szereplő bármely, időpontként értelmezhető numerikus, karakteres vagy ezek kombinációját tartalmazó kifejezésből a percet emeli ki, és azt egy 1 és 59 közötti egész számként adja meg. Ha az argumentum **Null**, a függvény ugyancsak **Null** eredményt ad.

MIRR()

Szintaxis: **MIRR(** ***values*( ), *finance_rate*, *reinvest_rate*** **)**

Ezzel a **MIRR( )** (*Modified Internal Rate of Return*) pénzügyi függvénnyel egy rendszeres időközönként pénzforgalmat lebonyolító számlán adott időtartam alatti két eltérő kamatszint mellett elért relatív kamatmértékét lehet meghatározni.

- A névvel hivatkozott ***values*( )** argumentum egy adattömb, amelyre az **IRR( )** függvénynél leírtak érvényesek.
- A ***finance_rate*** a kölcsönként felvett (az adattömb negatív elemeihez tartozó) összeg után fizetendő kamatot jelöli.
- A ***reinvest_rate*** a kölcsönadott, ismét befektetett (pozitív elemek) összeg után kapott kamatot jelöli.

A ***finance_rate*** és ***reinvest_rate*** argumentumokat decimális értékként, azaz 12% kamat esetén 0.12 alakban kell megadni.

Month()

Szintaxis: **Month(** *dátum* **)**

A függvény a *dátum* argumentum bármely, dátumként értelmezhető karakteres, numerikus vagy ezek kombinációját tartalmazó kifejezéséből a hónapot emeli ki, és azt egy 1 és 12 közötti egész számként adja meg. Ha az argumentum **Null**, a függvény ugyancsak **Null** eredményt ad.

MsgBox()

Szintaxis: **MsgBox(** ***prompt*** [, ***buttons***] [, ***title***] [, ***helpfile*, *context***])

A függvény egy karakteres kifejezésében meghatározott üzenetet egy párbeszédpanelen megjeleníti, majd addig várakozik, amíg a felhasználó egy nyomógombot nem használ. A függvény értéke a használt gomb azonosítója lesz.

- A párbeszédpanelen megjelenő üzenet karakterláncát a függvény névvel hivatkozott ***prompt*** argumentumában kell megadni. Az üzenet a használt karaktertípus szélességétől függően kb. 1024 karakternyi hosszú lehet. A többsoros üzeneteket az egyes sorok közé a kocsivisssza, azaz **Chr**(13) és a soremelés, azaz **Chr**(10) karakterek, illetve ezek kombinációjának **Chr**(13) **&** **Chr**(10) elhelyezésével lehet tördelni.

A párbeszédpanel megjelenését további névvel hivatkozott, de feltételes argumentumokkal lehet befolyásolni.

- A ***buttons*** argumentum a párbeszédpanelen megjeleníteni kívánt nyomógombo-

kat és azok típusát, a használt ikonok stílusát, az alapértelmezett gombokat és a párbeszédpanel modalitását kijelölő értékek összegét tartalmazó numerikus kifejezés. Alapértelmezett értéke 0, lehetséges beállításait pedig a következő táblázatok tartalmazzák. A következő négy csoportból csoportonként egyszerre csak egy érték vagy szimbolikus konstans használható. Ha az argumentumból hiányzik a ***buttons*** argumentum, azt a függvény nullának tekinti.

A párbeszédpanel nyomógombjainak száma és típusa:

Ha a szimbolikus konstans:	Értéke:	Megjelenő nyomógombok:
vbOKOnly	0	Csak az OK gomb
vbOKCancel	1	Az OK és Cancel gomb
vbAbortRetryIgnore	2	Az Abort, Retry és Ignore gombok
vbYesNoCancel	3	A Yes, No és Cancel gombok
vbYesNo	4	A Yes és No gombok
vbRetryCancel	5	A Retry és Cancel gombok

A párbeszédpanel ikonjainak stílusa:

Ha a szimbolikus konstans:	Értéke:	A megjelenő ikon:
vbCritical	16	A kritikus üzenet ikon
vbQuestion	32	A "?" ikon
vbExclamation	48	A "!" ikon
vbInformation	64	Az "i" ikon

Az alapértelmezett gomb:

Ha a szimbolikus konstans:	Értéke:	Az alapértelmezett nyomógomb:
vbDefaultButton1	0	Az első gomb
vbDefaultButton2	256	A második gomb
vbDefaultButton3	512	A harmadik gomb

A párbeszédpanel modalitása:

Ha a szimbolikus konstans:	Értéke:	A párbeszédpanel modalitása:
vbApplicationModal	0	Alkalmazási modalitás, azaz az éppen futó alkalmazás további futásának feltétele, hogy a felhasználó a párbeszédpanelen választ adjon.
vbSystemModal	4096	Rendszermodalitás, azaz minden alkalmazás futása mindaddig felfüggesztődik, amíg a felhasználó a párbeszédpanelen valamilyen választ nem ad.

- A ***title*** argumentumban a párbeszédpanel címsorában megjeleníteni kívánt szöveget tartalmazó karakteres kifejezést adhatjuk meg, ennek elhagyásakor a címsor az alkalmazás nevét tartalmazza.

- A ***helpfile*** és a ***context*** argumentumok használatával súgót rendelhetünk a függvénnyel megjelenítendő párbeszédpanelhez. A ***helpfile*** karakteres kifejezés, amelyben az adott párbeszédpanelhez tartozó súgót, magyarázószöveget tároló fájl nevét kell megadni, a ***context*** pedig numerikus kifejezés, amely a súgófájlon belüli téma számát jelöli ki. A ***helpfile*** és a ***context*** argumentumokat csak együtt lehet megadni vagy elhagyni. Ha megadottak, a ***helpfile*** argumentum adja meg az adott párbeszédpanelhez tartozó magyarázó szöveget tartalmazó súgófájl nevét, teljes elérési útját, a ***context*** numerikus argumentum pedig a fájlon belül a megfelelő téma számát. Ilyen esetben a felhasználó a `F1` funkcióbillentyű leütésével jelenítheti meg a ***context*** által kijelölt súgó témakört.

A párbeszédpanelen megjelenített nyomógombok közül valamelyiket működtetetve a **MsgBox( )** függvény a következő eredményt adja vissza:

Ha a szimbolikus konstans:	Értéke:	A lenyomott gomb:
vbOK	1	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetry	4	Retry
vbIgnore	5	Ignore
vbYes	6	Yes
vbNo	7	No

Ha a párbeszédpanel Cancel gombot tartalmaz, az `ESC` billentyű leütése azonos hatású a Cancel gomb lenyomásával. Ha a párbeszédpanel Help gombot tartalmaz, a panelhez témaérzékeny súgó tartozik.

Az argumentumlista bármelyik feltételes tagja elhagyható, azonban az előtte szereplő elválasztó vesszőt ki kell tenni.

Now

Szintaxis: **Now**

Ezzel az argumentum nélküli függvénnyel a rendszeren belül érvényes dátumot és időt lehet kiolvasni. Az eredmény egy valós szám, amelyben a tizedespont előtti rész a dátumot, a tizedespontot követő rész pedig az időpontot jelzi.

NPer( )

Szintaxis: **Nper(** ***rate, pmt, pv*** **[,** ***fv*** **[,** ***type*** **]])**

Ez az **Nper( )** (*Number of Periods*) olyan gazdasági függvény, amelyet adott, állandó kamat és részletfizetési összeg mellett egy meghatározott kölcsöntörlesztéshez vagy megtakarítás eléréséhez szükséges pénzügyi műveletek (ki- vagy befizetések) számát

adja meg.

A névvel hivatkozott argumentumok értelmezése a ***rate***, ***pmt***, ***pv*** és ***type*** esetén az **FV**, az ***fv*** esetén pedig az **Ipmt** függvénynél használt, azonos nevű argumentumokéval azonos.

NPV( )

Szintaxis: **NPV(** ***rate***, ***values*****( ))**

Ezzel az **NPV( )** (*Net Present Value*) pénzügyi függvénnyel egy periodikus, rendszeres időközönkénti, de változó részösszegű befektetés adott diszkontkamat figyelembevételével számított, jelenleg érvényes nettó értékét határozhatjuk meg.

- A ***rate*** névvel hivatkozott argumentum egy adott periódusra vonatkozó diszkontkamat decimális alakban adott értéke.
- A ***values*****( )** argumentum egy adattömb, amelyre az **IRR( )** függvénynél leírtak érvényesek.

A jelenlegi netto érték a jövőbeli befizetések alapján kerül számításra. A pénzügyi műveletek sora az első befizetés előtt egy periódussal kezdődik, és az utolsó befizetéssel fejeződik be. Ha egy befizetés az első periódus kezdetén történik, azt az **NPV** függvény eredményéhez kell hozzáadni, nem szabad a ***values*****( )** argumentumban adott befizetési értékek között szerepeltetni.

Ha a befektetési részösszegek állandók, a jelenleg érvényes nettó értéket az **NPV( )** függvényhez nagyon hasonló **PV( )** függvénnyel számíthatjuk.

Oct( )

Szintaxis: **Oct(** *szám* **)**

A függvény az argumentumban lévő bármely érvényes numerikus vagy számként értelmezhető karakteres kifejezés legközelebbi egész számra kerekített értékének oktális, azaz nyolcas számrendszerbeli alakját egy legfeljebb 11 oktális karakterből álló karakterláncként adja meg. Ha a *szám* argumentum **Null**, az eredmény is **Null**, ha pedig **Empty**, a függvény eredménye 0 lesz.

A kifejezésekben közvetlenül használandó oktális értékeket az **&O** előtaggal kell megkülönböztetni az egyéb karakterláncoktól.

OpenDatabase

Szintaxis: **OpenDatabase**

A függvény megnyit egy adott adatbázist és eredményül egy objektum adattípusának értékét adja vissza az adatbázisnak.

A **Workspace** objektummal inkább az **OpenDatabase** metódus használata javasolt.

Partition( )

Szintaxis: **Partition(** ***number***, ***start***, ***stop***, ***interval*** **)**

Ez a függvény egy számnak egy adott számtartományon belüli helyzetét megadó karakterláncot ad eredményül.

- A vizsgálandó számot a névvel hivatkozott ***number*** argumentumban kell megadni.
- A tartomány kezdetét a nullánál nagyobb értékű ***start***, a tartomány végét pedig a ***start*** értékénél nagyobb ***stop*** argumentumban kell elhelyezni.
- A ***start*** és ***stop*** által kijelölt tartomány felosztási sűrűségét az egynél nem kisebb ***interval*** argumentum értéke határozza meg. Ha a tartomány nem az ***interval*** egész számú többszöröse, vagyis nem maradék nélkül osztható fel, az utolsó tartomány csak a maradéknak megfelelő hosszúságú lesz.

Minden argumentumnak egész számnak kell lennie, ha bármelyik **Null**, az eredmény is **Null** lesz.

Az eredmény a számnak a felosztott számtartományon belül helyét az aktuális ***interval*** megjelölésével adja meg, például a ***start*** = 10, a ***stop*** = 90 és az ***interval*** 10, a ***number*** következő értékeinél a következő alakú eredményeket kapjuk:

Ha a ***number*** értéke:	Az eredmény:
-3	" :-3"
4	" 0: 4"
45	" 40:49"
92	" 92: "

Mint látható, az eredmény jobb és bal oldalán mindig a ***stop*** nagyságrendjének megfelelőnél eggyel több karakterpozíciót kapunk. A függvény használatára további példák találhatók a Súgó ezen témakörénél.

Ha az ***interval*** értéke 1, a függvény eredménye a többi argumentum értékétől függetlenül mindig ***number*** :***number*** alakú lesz.

A függvény leggyakrabban kiválasztások lekérdezési feladataihoz használható.

Pmt( )

Szintaxis: **Pmt(** ***rate***, ***nper***, ***pv*** [, ***fv*** [, ***type***]] **)**

Ezzel a **Pmt( )** (*Payment*) gazdasági függvénnyel egy adott kölcsön vagy megtakarítás meghatározott időtartam alatti, állandó kamatfeltételek mellett egyenlő részletekben történő fizetése esetén érvényes részösszeg határozható meg.

A függvény névvel hivatkozott argumentumainak értelmezése teljesen azonos az **FV( )** és **Ipmt( )** függvények azonos nevű argumentumainál leírtakkal.

PPmt( )

Szintaxis: **PPmt(** ***rate, per, nper, pv*** **[,** ***fv*** **[,** ***type*** **]])**

Ezzel a **PPmt( )** (*Principal Payment*) gazdasági függvénnyel egy adott kölcsön vagy megtakarítás meghatározott időtartam alatti tőkefizetési értéke határozható meg állandó kamatfeltételek és egyenlő részletekben történő fizetés esetén.

A függvény névvel hivatkozott argumentumainak értelmezése teljesen azonos a kamatfizetési érték meghatározásra használható **IPmt( )** függvény azonos nevű argumentumainál leírtakkal.

PV( )

Szintaxis: **PV(** ***rate, nper, pmt*** **[,** ***fv*** **[,** ***type*** **]])**

Ezzel az **FV( )** függvényhez hasonló **PV( )** (*Present Value*) gazdasági függvénnyel egy adott futamidő végére az azonos részösszegű, rendszeres megtakarítás vagy kölcsöntörlesztés állandó kamat mellett elért összegének jelenlegi értéke határozható meg.

A függvény névvel hivatkozott argumentumainak értelmezése azonos a **PV( )** és **Ipmt( )** függvény azonos nevű argumentumainál leírtakkal.

QBColor( )

Szintaxis: **QBColor(** *szín* **)**

A függvény eredménye a *szín* argumentum 1 és 15 közötti színértékének megfelelő RGB színkód, amelyben a legkisebb helyértékű byte-tal kezdődően a piros, zöld és kék összetevők szerepelnek. Az egyes színértékekhez tartozó színek a következők:

Ha a *szín* értéke:	A szín:	Ha a *szín* értéke:	A szín:
0	fekete	8	szürke
1	kék	9	világoskék
2	zöld	10	világoszöld
3	türkizkék	11	világos türkiz
4	piros	12	élénkpiros
5	bíborvörös	13	halványbíbor
6	sárga	14	halványsárga
7	fehér	15	intenzív fehér

A Basic más verziói (például a Visual Basic for MS-DOS vagy a Basic Compiler) ugyancsak az itt felsorolt színértékeket használják.

Rate( )

Szintaxis: **Rate(** ***nper, pmt, pv*** [, ***fv*** [, ***type*** [, ***guess***]]])

Ezzel a pénzügyi függvénnyel egy rendszeres, adott időtartamig tartó befizetéssel elért összeghez (megtakarítás vagy törlesztés) tartozó átlagos kamatláb számítható.

A függvény névvel hivatkozott argumentumainak értelmezése a ***guess*** kivételével teljesen azonos a **PV( )** függvény azonos nevű argumentumainál leírtakkal.

- A feltételes ***guess*** argumentumban egy becsült kamatszintet lehet megadni, amelyből a függvény legfeljebb húsz iterációs lépéssel kísérli meghatározni a tényleges értéket. Ha húsz iterációs lépés után sem jobb az eredmény pontossága mint 0.00001%, a számítás sikertelen lesz. A ***guess*** alapértelmezetten 0.1, azaz 10%. Ha a függvény az iterációs eljárás végén nem talál megfelelő végeredményt, más becsléssel a függvényt újra kell indítani.

RGB( )

Szintaxis: **RGB(** ***red, green, blue***)

A függvény eredménye egy, a névvel hivatkozott argumentumok értékével meghatározott színtől függő egész szám lesz. A ***red*** (piros), ***green*** (zöld) és ***blue*** (*kék*) argumentumok 0 és 255 közötti számok, és az adott színben az egyes színösszetevők arányát adják meg. A 255-nél nagyobb értékű argumentumot az **RGB** függvény 255-nek tekinti.

Az egyes standard színek RGB értékeinek hexadecimális alakját és RGB összetevőit a következő táblázat tartalmazza:

Szín:	RGB érték:	piros:	zöld:	kék:
fekete	&H00	0	0	0
kék	&HFF0000	0	0	255
zöld	&HFF00	0	255	0
türkizkék	&HFFFF00	0	255	255
piros	&HFF	255	0	0
bíbor	&HFF00FF	255	0	255
sárga	&HFFFF	255	255	0
fehér	&HFFFFFF	255	255	255

A színmegadást elfogadó alkalmazási metódusok és tulajdonságok a színt RGB színértékként értelmezhető számként várják.

Right( )

Szintaxis: **Right(** ***string, length***)

A függvényre vonatkozó tudnivalók teljesen azonosak a **Left( )** függvénynél leírtak-

kal, azzal az értelemszerű különbséggel, hogy a ***length*** argumentumban megadott számú karakter a ***string*** argumentumban adott karakteres kifejezés jobb oldaláról kerül az eredménybe.

RightB( )

Szintaxis: **RightB(** ***string, length*** **)**

A függvényre vonatkozó tudnivalók teljesen azonosak a **LeftB( )** függvénynél leírtakkal, azzal az értelemszerű különbséggel, hogy a ***length*** argumentumban megadott számú byte a ***string*** argumentumban adott karakteres kifejezés jobb oldaláról kerül az eredménybe.

Rnd( )

Szintaxis: **Rnd** [(*szám*)]

A függvény 0 és 1 közötti ál-véletlenszámot állít elő. A feltételes *szám* argumentum bármely érvényes numerikus kifejezés lehet, értéke alapértelmezetten > 0. A véletlen szám előállításának módját a *szám* értéke a következők szerint határozza meg:

Ha a *szám*:	A függvény eredménye:
< 0	Mindig ugyanazon, a függvény magjaként a *szám* értékével előállított véletlen szám
> 0	A sorrendben következő véletlen szám
= 0	Az előzőleg előállított véletlen szám

A véletlen számok képzését végző algoritmusból következően a függvény azonos kiindulási magérték esetén minden egyes meghíváskor ugyanazon véletlenszámsorozatot szolgáltatja, mivel a véletlen számok egymás utáni előállításához az előző véletlen számot használja magértékként. "Véletlenebb" számsorozat előállításához az **Rnd( )** meghívása előtt a véletlenszám-generátort az argumentum nélküli **Randomize** utasítással kell inicializálni, amely a magértéket a rendszer órájának aktuális értékéből állítja elő.

Ismétlődő véletlenszám-sorozat előállításához közvetlenül a **Randomize** numerikus argumentummal történő használata előtt kell az **Rnd( )** függvényt negatív argumentummal meghívni, ugyanis a **Randomize** utasítás még ugyanazon értékű argumentummal sem eredményezi ugyanazon számsort.

RTrim( )

Szintaxis: **RTrim(** *karakterlánc* **)**

A függvény eredménye a *karakterlánc* argumentumban szereplő bármely érvényes

karakteres_kifejezés jobb oldali (azaz követő) szóközök nélküli másolata. Ha az argumentum **Null** típusú, az eredmény is **Null** lesz.

Second()

Szintaxis: **Second(** *idő* **)**

A függvény az *idő* argumentumban lévő bármely érvényes, időpontként értelmezhető numerikus vagy karakteres kifejezésből, vagy ezek kombinációjából csak a másodpercet jelző részt emeli ki, és eredményül egy annak megfelelő, 0 és 59 közötti egész számot ad meg. Ha a *szám* argumentum **Null**, függvény eredménye is **Null** lesz.

Seek()

Szintaxis: **Seek(** *fájlszám* **)**

A függvényben a *fájlszám* argumentumaként az **Open** utasításban a fájl megnyitására használt számot használva, a függvény a fájlon belül éppen érvényes írási/olvasási pozícióval tér vissza. A **Random** módban megnyitott fájloknál az eredmény a következő írható vagy olvasható rekord, a **Binary**, **Output**, **Append** és **Input** módban megnyitottaknál pedig az a byte lesz, amelyre a következő művelet irányul. Az eredmény 1 és 2147483647 (2^{31}-1) közötti szám lehet. A fájlon belüli pozíció számozása az első bytetól kezdődik.

Sgn()

Szintaxis: **Sgn(** *szám* **)**

A függvény a *szám* argumentumban megadott bármilyen érvényes numerikus kifejezés eredményének előjelétől függően a következő eredményt adja:

Ha a *szám*:	Az eredmény:
< 0	-1
> 0	1
= 0	0

Shell()

Szintaxis: **Shell(** ***pathname*** [, ***windowstyle***] **)**

A függvény a ***pathname*** argumentumban az elérési úttal, esetleg a futtatáshoz szükséges egyéb argumentumokkal, utasítássor-kapcsolóval és kapcsolódó dokumentumokkal kiegészített nevével kijelölt futtatható fájlok futtatására alkalmas.

A feltételes ***windowstyle*** argumentumban a programot futtató ablak stílusát írhatjuk

elő a következő táblázatnak megfelelően:

Ha a szimbolikus konstans:	Értéke:	Az ablakstílus:
vbHide	0	Az ablak rejtett, a fókusz a rejtett ablaknak van átadva.
vbNormalFocus	1	Az ablak az eredeti méretre és pozícióba van visszaállítva, a fókusz ablaké.
vbMinimizedFocus	2	Az ablak ikonméretűre csökkentett, a fókusz az ikonméretű ablaké. Alapértelmezett.
vbMaximizedFocus	3	Az ablak teljes méretű, a fókusz az ablaké.
vbNormalNoFocus	4	Az ablak az előző méretre és pozícióra van visszaállítva, az éppen aktív ablak továbbra is aktív marad.
vbMinimizedNoFocus	6	Az ablak ikonméretű, az éppen aktív ablak továbbra is aktív marad.

Az adott program sikeres lefutása után a **Shell( )** függvény eredményeként az elindított program egyedi azonosítószámát (ID) kapjuk vissza. Ha a függvény nem tudja az argumentumban definiált programot elindítani, hibajelzést kapunk.

A **Shell( )** függvény a többi programot aszinkron módon, vagyis más programoktól függetlenül futtatja, ezeket a programokat nem lehet egymáshoz kapcsolni.

Sin( )

Szintaxis: **Sin(** *szám* **)**

A függvény a *szám* argumentumban adott, egy szög radiánban kifejezett argumentumaként értelmezhető számhoz vagy numerikus kifejezés eredményéhez tartozó, -1 és 1 közé eső szinuszértéket adja meg. A fok radiánra történő átszámítása az 1 fok = $\pi/180$ = 0,0174532925199433 radián azonosság szerint történhet.

Space( )

Szintaxis: **Space(** *szám* **)**

A függvény a *szám* argumentumban adott számú üres karaktert (betűközt) tartalmazó karaktersorozatot ad vissza.

Spc( )

Szintaxis: **Spc(** *szám* **)**

Az értelemszerűen csak a **Print** metódussal vagy **Print#** utasítással használható függvénnyel egy listában a *szám* argumentumban adott számú szóközzel léptethető to-

vább az éppen érvényes kiírási, nyomtatási pozíció.

A függvény a *szám* értékétől, a kimeneten a soron belüli éppen érvényes pozíciótól és a sorszélességtől függően számítja ki az új pozíciót, illetve szükség esetén a soremelések és az új soron belül szükséges szóközök számát:

- ha a *szám* kisebb, mint a sor szélessége, a kiírás közvetlenül az adott számú szóköz után folytatódik.
- az argumentumban a sor szélességénél nagyobb értékű *szám* esetén a függvény az éppen érvényes pozícióhoz a (*szám* **Mod** *sorszélesség*) értékének hozzáadásával határozza meg az új pozíciót. Ha az így számított pozíció és sor szélessége közötti különbség kisebb mint a *szám* (vagy a *szám* Mod *sorszélesség*) értéke, az **Spc( )** függvény a következő sornak a jelenlegi pozíció és a (*szám* — *sorszélesség*) összegének megfelelő számú pozícióban folytatja a kiírást.

Az arányos (proporcionális) betűkkel végzett kiíratáskor az **Spc( )** függvény az adott betűtípus összes karaktereinek pontban mért szélességeinek átlagát tekinti a szóköz szélességének. Ilyen esetben nincs pontos összefüggés a kiírt karakterek száma és az általuk elfoglalt állandó oszlopszélesség között, hiszen például a W karakter több helyet foglal el mint az l.

Az űrlapok és hasonló táblázatok esetén érdemes ellenőrizni, hogy a rendelkezésre álló hely elegendő-e a széles karakterek beírásához is.

Sqr()

Szintaxis: **Sqr(** *szám* **)**

A függvény a *szám* argumentumban adott pozitív szám négyzetgyökét határozza meg. Negatív érték hibajelzést okoz.

Str()

Szintaxis: **Str(** *numerikus_kifejezés* **)**

A függvény a *numerikus_kifejezés* argumentumban adott bármilyen numerikus kifejezést egyszerű, formattálatlan karakterlánccá alakítja. A pozitív számok átalakításakor az előjel helyén üres (szóköz) karakter jelenik meg.

A függvény elsősorban numerikus adatoknak dátumra, időpontra, pénzegységre vagy felhasználó által definiált formára történő konvertálására használható. A függvény tizedesjelként csak a pontot (.) ismeri fel, ezért a más tizedesjelet használó területi beállítások lehetősége esetén a **Cstr( )** függvényt kell használni a számok karakterlánccá alakítására.

StrComp()

Szintaxis: **StrComp(** ***string1***, ***string2*** [, ***compare***] **)**

A függvény a névvel hivatkozott, bármely érvényes karakteres kifejezést tartalmazó ***string1*** és ***string2*** argumentumok összehasonlításának eredményétől függő értéket ad vissza.

A feltételes ***compare*** argumentum a ***string1*** és a ***string2*** összehasonlítás típusát írja elő:

- Ha a ***compare*** hiányzik, az összehasonlítás az **Option Compare** utasítással előírt módon történik.
- Ha a ***compare*** = 0 (vagyis az alapértelmezett érték), az összehasonlítás binárisan történik.
- Ha a ***compare*** = 1, az összehasonlítás szövegek összehasonlításaként, a nagy- és kisbetűs karaktereket azonosnak tekintve történik.
- Ha a ***compare*** = **Null**, hiba generálódik.
- Ha a ***compare*** = egy **Field** objektum **CollatingOrder** tulajdonságának értéke, egy adatbázis értékeinek rendezése vagy összehasonlítása történik, ugyanolyan módon, ahogy azt az adatbázis végezné.

Az összehasonlítás eredménye a következő lehet:

Ha az összehasonlítás:	Az eredmény:
string1 < ***string2***	-1
string1 = ***string2***	0
string1 > ***string2***	1
string1 = **Null** vagy ***string2*** = **Null**	**Null**

StrConv()

Szintaxis: **StrConv(** ***string***, ***conversion*** **)**

A függvény a névvel hivatkozott ***string*** argumentumban adott bármilyen karakteres kifejezést a ***conversion*** argumentumban adott értékkel meghatározott konverziós módnak megfelelően átalakított alakban adja vissza.

A névvel hivatkozott ***conversion*** argumentumban a következő értékek használhatók:

Ha a szimbolikus konstans:	Értéke:	Hatása:
vbUpperCase	0	A karakterláncot kisbetűs alakra konvertálja
vbLowerCase	2	A karakterláncot nagybetűs alakra konvertálja
vbProperCase	3	A karakterlánc minden szavának első karakterét nagybetűre konvertálja
vbUnicode	64	A karakterláncot a rendszer alapértelmezett kódlapjának megfelelően Unicode-ba konvertálja

vbFromUnicode	128	A karakterláncot Unicode-ból a rendszer alapértelmezett kódlapjának megfelelően konvertálja

Az argumentum lehetséges értékei közül — az egymást kizárók kivételével — egyszerre több is megadható.

A ***string*** karakterláncában a következő szóközök érvényesek:

Null	Chr$(0)
Vízszintes tabulátor	Chr$(9)
Soremelés	Chr$(10
Függőleges tabulátor	Chr$(11)
Lapdobás	Chr$(12)
Kocsivissza	Chr$(13)
Szóköz (területi beállítástól függő)	Chr$(32)

Megjegyezzük, hogy a Visual Basic távol-keleti és japán verzióiban további négy ***conversion*** beállítási lehetőség létezik, azonban azok más területi beállításnak megfelelő környezetben futásidejű hibát okoznak.

String()

Szintaxis: **String(** ***number***, ***character*** **)**

A függvény a névvel hivatkozott ***number*** argumentumban adott számú, azonos karakterből álló karakterláncot ad vissza. Az ismétlődő karaktert a ***character*** argumentumban egy karakterkóddal vagy egy karakteres kifejezés első karaktereként lehet megadni.

Akár a ***number***, akár a ***character*** argumentumban szereplő **Null** esetén az eredmény is **Null** lesz.

Ha a ***character*** argumentumban adott karakterkód értéke nagyobb lenne mint 255, a függvény azt a (***character*** **Mod** 256) kifejezés szerint alakítja érvényes karakterkóddá.

Switch( )

Szintaxis: **Switch(** *kifejezés-1*, *érték-1* [... [, *kifejezés-n*, *érték-n*]])

A függvény a *kifejezés, érték* értékpárokat tartalmazó argumentumlista **Variant** típusú *kifejezés1*, *kifejezés-2*, ... *kifejezés-n* elemeit azok természetes sorrendjében (balról-jobbra) kiértékeli, majd az összes kifejezés kiértékelése után az első olyan kifejezéshez tartozó *eredmény* értékét vagy kifejezését adja vissza, amelynél a kiértékelés eredménye **True** volt.

Ha nincs **True** értékű kifejezés, az eredmény **Null** lesz. Ugyancsak **Null** eredményt kapunk, ha a **True** eredményű kifejezéshez tartozó eredmény **Null**. Hiányos, nem azonos számú kifejezést és eredményt tartalmazó argumentumlista futásidejű hibát okoz.

A függvény az összes kifejezést kiértékeli, ezért itt is számítani kell az **IIf( )** függvénynél említett mellékhatásra.

SYD( )

Szintaxis: **SYD(** ***cost, salvage, life, per*** **)**

A **SYD( )** (*Summ-of-Years' Depreciation*) pénzügyi függvény egy vagyontárgynak egy adott periódus alatti értékcsökkenését határozza meg.

Az argumentumban szereplő argumentumok értelmezése teljesen azonos a **DDB( )** függvénynél leírtakkal, mindössze a számítási algoritmusban van eltérés.

Tab( )

Szintaxis: **Tab[** (*szám*) **]**

Az értelemszerűen csak a **Print** metódussal vagy **Print#** utasítással együtt használható függvénnyel a kiíratási pozíció a feltételes *szám* argumentumban adott értékéknek megfelelő számú oszloppal jobbra, új kiíratási pozícióba vihető.

A függvény a *szám* értékétől, a kimeneten a soron belüli éppen érvényes pozíciótól és a sorszélességtől függően számítja ki az új pozíciót, illetve szükség esetén a soremelések és az új soron belül szükséges oszlopok számát:

- Ha az adott sorban az éppen érvényes pozíció kisebb mint a *szám*, a kiírási pozíció az aktuális sorban a *szám* értékének megfelelő oszlopba kerül.
- Ha az adott sorban az éppen érvényes pozíció nagyobb mint a *szám*, a kiírási pozíció a következő sorba, a *szám* értékének megfelelő oszlopba kerül.
- Ha a *szám* kisebb mint 1, a függvény a kiíratást az aktuális sor első oszlopában folytatja.
- Ha a *szám* nagyobb értékű a sor szélességénél, a függvény a következő sorban, a *szám* Mod *sorszélesség* értékének megfelelő oszlopban folytatja a kiírást.
- Ha a *szám* argumentum hiányzik, az új pozíció a következő kiíratási zóna elejére kerül.

A fájlok **Print#** utasítással történő kiíratásakor a jobb szélső kiíratási pozíció a kimeneti fájl éppen érvényes — a **Width#** utasítással beállítható — szélessége lesz.

A **Tab( )** függvényt a **Print** metódussal használva a kiíratási felület egyenlő szélességű oszlopokra osztódik. Az oszlopok szélessége az adott betűtípus összes karaktereinek pontban mért szélességeinek átlaga lesz. Ilyen esetben nincs pontos összefüggés a kiírt karakterek száma és az általuk elfoglalt állandó oszlopszélesség között, hiszen például a W karakter több helyet foglal el mint az l.

Az űrlapok és hasonló táblázatok esetén érdemes ellenőrizni, hogy a rendelkezésre álló hely elegendő-e a széles karakterek beírásához is.

Tan()

Szintaxis: **Tan(** *szám* **)**

Ez a függvény a *szám* argumentumban szereplő számot vagy bármely érvényes numerikus kifejezés eredményét a tangens szögfüggvény radiánban kifejezett argumentumaként értelmezve, a hozzá tartozó tangens értékét adja meg. A fok 1 fok = 0,0174532925199433 radián arány alapján számítható át radiánra. Az eredmény túl nagy szám esetén ($\pi/2$ és $3*\pi/2$ környékén) esetleg *Owerflow* hibát okozhat.

Time

Szintaxis: **Time**

Ez az argumentum nélküli függvény a rendszerben érvényes időpont **Date** értékét adja vissza. Az időpont beállítása a **Time** utasítással történik.

Timer

Szintaxis: **Timer**

Ez az argumentum nélküli az éjfél (12:00 AM) óta eltelt másodpercek számát adja meg. A **Timer( )** függvényt például a **Randomize** utasítás használja a véletlen számok előállítására.

TimeSerial()

Szintaxis: **TimeSerial(** ***hour***, ***minute***, ***second*** **)**

A függvénnyel a névvel hivatkozott argumentumok értékét megfelelően — a ***hour*** 0 és 12 (vagy 0 és 23) közötti, a ***minute*** és ***second*** pedig 0 és 59 közötti számként vagy bármely érvényes numerikus kifejezésekként — megadva, egy **Date** időpontot kapunk eredményül.

A **TimeSerial( )** argumentumai relatív, például `TimeSerial(8-3, 45-6, 59-43)` alakban is megadhatók.

Akár a közvetlen, akár a relatív argumentumok használata esetén a természetes értékhatárokat túllépő érték automatikusan a következő nagyobb időegységre konvertálódik. Bármelyik, az **Integer** számtartományból kilépő argumentum, valamint a megengedett időtartományból kilépő eredmény is hibát okoz.

TimeValue( )

Szintaxis: **TimeValue**(*idő*)

A függvény az *idő* argumentumban adott bármilyen érvényes, időpontként értelmezhető (általában karakteres) kifejezésként megadott időpontot **Date** értékként ad vissza.

Az *idő* 12 vagy 24 órás rendszerben adott, például a 17:33:15 vagy 5:33:15 PM alakú időpontot egyaránt tartalmazhat. Ha a kifejezésben dátumként értelmezhető elem is van, azt a függvény figyelmen kívül hagyja, azonban a hibás dátumalak hibát okoz.

Trim( )

Szintaxis: **Trim**(*karakterlánc*)

A függvény eredménye a *karakterlánc* argumentumban szereplő bármely érvényes karakteres_kifejezés jobb és bal oldali (azaz a bevezető és a követő) szóközök nélküli másolata. Ha az argumentum **Null** típusú, az eredmény is **Null** lesz.

TypeName( )

Szintaxis: **TypeName**(*változónév*)

A függvény a *változónév* argumentumban adott változó adattípusát tartalmazó karakterláncot ad vissza:

Ha az eredmény:	A változó típusa:
objektumtípus	Egy adott típusú OLE Automatizmus objektum
Byte	**Byte**
Integer	**Integer** — egész
Long	**Long** — hosszú egész
Single	**Single** — egyszeres pontosságú lebegőpontos
Double	**Double** — dupla pontosságú lebegőpontos
Currency	**Currency** — pénzügyi
Date	**Date** — dátum
String	**String** — karakterlánc
Boolean	**Boolean** — logikai
Error	**Error** — hiba
Empty	**Empty** — inicializálatlan
Null	**Null** — érvénytelen
Object	OLE Automatizmust támogató objektumtípus
Unknown	Ismeretlen típusú, OLE Automatizmust támogató objektumtípus
Nothing	Objektumra nem hivatkozó objektumváltozó

Ha a *változónév* egy tömb, a függvény által visszaadott karakterlánc egy üres zárójel-párral egészül ki, például egy integer tömb esetén az eredmény `Integer( )` alakú lesz.

UBound()

Szintaxis: **UBound(** *tömbnév* [, *dimenzió*])

A függvény a *tömbnév* argumentummal kijelölt nevű adattömbnek a *dimenzió* argumentumban adott számnak megfelelő dimenziója számára definiált legnagyobb indexet adja vissza, egyéb jellemzői és használata értelemszerűen azonos az **Lbound( )** függvénynél leírtakkal.

Az **Lbound( )** és **Ubound( )** függvények együttesen egy tömb méretének meghatározására használhatók.

UCase()

Szintaxis: **UCase(** *karakterlánc*)

A függvény a *karakterlánc* argumentumban adott bármely érvényes karakteres kifejezést nagybetűs alakban — a kisbetűs karaktereket nagybetűs alakkal helyettesítve, az összes többi karaktert változatlanul hagyva — adja vissza. Ha a *karakterlánc* karakteres kifejezése **Null**, az eredmény is **Null** lesz.

Val()

Szintaxis: **Val(** *karakterlánc*)

A függvény a *karakterlánc* argumentumában adott bármely érvényes, számként értelmezhető karakteres kifejezésben szereplő számot adja vissza.

Az átalakítás az első, nem számjegy karakternél leáll. A függvény a számjegyek közötti szóközt, tabulátort és soremelést nem adja vissza, valamint nem ismeri fel az olyan, általában számok között előforduló szimbólumokat mint a "," és "$" sem. Mivel csak a tizedespontot "." fogadja el érvényes tizedesjelnek, a más tizedesjelet használó területi beállítások esetén a **Val( )** helyett a **Cdbl( )** függvényt kell alkalmazni.

Ugyanakkor a **Val( )** felismeri az oktális rendszerhez tartozó **&O** és a hexadecimális rendszerhez tartozó **&H** előtagokkal jelölt számokat, így azok decimális alakra konvertálásához is felhasználható.

VarType( )

Szintaxis: **VarType(** *változónév* **)**

A függvény eredményül egy, a *változónév* argumentumban — a felhasználó által definiált típus kivételével — megadott bármely érvényes változó al-(belső) adattípusára jellemző számot ad vissza. Az eredmény értelmezése a következő:

Eredmény:	Szimbolikus konstans:	A változó adattípusa:
0	**vbEmpty**	**Empty** — inicializálatlan
1	**vbNull**	**Null** — érvénytelen
2	**vbInteger**	**Integer** — egész
3	**vbLong**	**Long** — hosszú egész
4	**vbSingle**	**Single** — egyszeres pontosságú lebegőpontos
5	**vbDouble**	**Double** — dupla pontosságú lebegőpontos
6	**vbCurrency**	**Currency** — pénzügyi
7	**vbDate**	**Date** — dátum
8	**vbString**	**String** — karakterlánc
9	**vbObject**	OLE Automatizmus objektum
10	**vbError**	**Error** — hiba
11	**vbBoolean**	**Boolean** — logikai
12	**vbVariant**	**Variant** — csak **Variant** tömbnél
13	**vbDataObject**	Nem OLE Automatizmus objektum
17	**vbByte**	**Byte**
8192	**vbArray**	**Array** — tömb

A függvény a **vbArray** értéket sohasem egymagában adja vissza, hanem mindig valamely másik — az adott tömb típusát is kijelölő — értékhez van hozzáadva, például a 8194 eredmény egy **vbInteger** + **vbArray**, azaz egész számokból álló tömböt határoz meg. A **vbVariant** értéke csak a **vbArray** értékkel együtt fordul elő, jelezve, hogy a *változónév* argumentum **Variant** tömböt jelöl.

Weekday( )

Szintaxis: **Weekday(** ***date*** **[,** ***firstdatofweek*** **])**

A függvény a névvel hivatkozott ***date*** argumentumban szereplő, dátumként értelmezhető numerikus, karakteres vagy ezek kombinációját tartalmazó kifejezésből számított dátumból a hét napját adja meg. Ha a ***date*** argumentum **Null**, az eredmény is **Null** lesz.

A függvényben a feltételes ***firstdayofweek*** argumentum egy beépített állandó, amely a hét első napjaként használandó napot adja meg:

Ha a szimbolikus konstans:	Értéke:	A hét első napja:
vbUseSystem	0	Ha létezik, az alkalmazás beállítása, különben az NLS API beállítása szerinti
vbSunday	1	Vasárnap (alapértelmezett)
vbMonday	2	Hétfő
vbTuesday	3	Kedd
vbWednesday	4	Szerda
vbThursday	5	Csütörtök
vbFriday	6	Péntek
vbSaturday	7	Szombat

A függvény által visszaadott eredmény értelmezés a következő:

Eredmény értéke:	Szimbolikus konstansa:	A *date* napja:
1	**vbSunday**	Vasárnap
2	**vbMonday**	Hétfő
3	**vbTuesday**	Kedd
4	**vbWednesday**	Szerda
5	**vbThursday**	Csütörtök
6	**vbFriday**	Péntek
7	**vbSaturday**	Szombat

Year()

Szintaxis: **Year(** *dátum* **)**

A függvény a *dátum* argumentumában szereplő dátumként értelmezhető numerikus, karakteres vagy ezek kombinációját tartalmazó kifejezésből az évszámnak megfelelő, számot adja eredményül. Ha a *dátum* argumentum **Null**, az eredmény is **Null** lesz.

Utasítások

AppActivate

Szintaxis: **AppActivate** ***title*** [, ***wait***]

Az utasítással egy adott címsorú alkalmazási ablak aktiválható.

- Az ablak címsorát a névvel hivatkozott ***title*** argumentumának karakteres kifejezésében kell megadni. Ha a ***title*** argumentumban a **Shell( )** függvény által visszaadott feladatazonosító (ID) szerepel, az utasítás az adott alkalmazást aktiválja. Az aktiválás egyben erre az ablakra helyezi a fókuszt, függetlenül az adott ablak állapotától. Ha egy alkalmazási programnak több azonos című aktiválható

ablaka van, ezek közül a választás véletlenszerű lesz.

- A feltételes *wait* argumentum logikai értéke azt határozza meg, hogy a hívó alkalmazás egy másik eljárás aktiválásakor már a fókuszban legyen-e. A *wait* alapértelmezetten **False** értéke esetén a háttérben futó, nem fókuszban lévő alkalmazásból is aktiválható egy alkalmazás, míg **True** érték esetén az adott alkalmazás mindaddig várakozik, amíg csak az **AppActivate** utasítást tartalmazó alkalmazás fókuszba kerülésekor meg nem történik.

Az **AppActivate** utasítás a *title* argumentummal adott alkalmazást vagy ablakot attól függetlenül helyezi a fókuszba, hogy az éppen ikonállapotú vagy teljes méretű-e. A fókusz az aktív ablakból felhasználói fókuszáthelyező művelettel vagy az ablak bezárásával helyezhető át. Egy alkalmazás elindítása és az ablak stílusának beállítása a **Shell( )** függvénnyel történhet. Az éppen aktív alkalmazások meghatározása a *title* és az egyes futó alkalmazások címsorainak összehasonlításával lehetséges. Ha nincs egyezés, a *title* által megadott alkalmazás aktiválódik. Több egyező címsorú alkalmazás esetén esetleges, hogy melyik fog aktívvá válni.

Beep

Szintaxis: **Beep**

Ezzel az argumentumnélküli utasítással a program futása során a számítógép beépített hangszóróján át a hardvertől és a rendszer szoftvertől függő időtartamú és frekvenciájú figyelmeztető hangjelzés adható.

BeginTrans

Szintaxis:

Ez az argumentum nélküli — csak kompatibilitási okokból megmaradt — utasítás az egyetlen egységként tekintett adatbázisokban végzett módosítások sorozatának elvégzését, azaz egy tranzakciót indít el. **Workspace** objektumokkal a tranzakció indítására inkább a **BeginTrans** metódus használandó.

Call

Szintaxis: [**Call**] *eljárásnév* [(*argumentumlista*)]

Az utasítás a vezérlést adott nevű **Sub** eljárás, **Function** eljárás vagy DLL számára adja át.

- Az utasítás *eljárásnév* argumentumának karakteres kifejezése tartalmazza az eljárás vagy DLL nevét.
- Ezt az *argumentumlista* argumentumban a meghívott eljárás számára átadandó

változók, tömbök vagy kifejezések zárójelben felsorolt, egymástól vesszővel elválasztott listája követi. Tömb neve után csak üres zárójelpárt kell megadni. DLL meghívása esetén az *argumentumlista* tartalmazhatja az argumentumok átadási módját előíró **ByVal** vagy **ByRef** kulcsszavakat is.

Ha a kívánt eljárást a feltételes **Call** kulcsszóval hívjuk meg, az *argumentumlista* kötelezően zárójelbe kerül, míg a **Call** nélküli alakban történő használat esetén a zárójeleket is el kell hagyni.

A **Call** utasítás a belső vagy felhasználó által definiált függvény meghívásakor annak eredményét nem adja vissza.

ChDir

Szintaxis: **ChDir** *elérési_út*

Az utasítással új könyvtárat lehet kijelölni. Az *elérési_út* argumentumban szereplő karakteres kifejezés a továbbiakban alapértelmezettnek tekintendő könyvtár nevét a következő alakban tartalmazhatja:

[*meghajtó* : \ ] *könyvtár1* [[*könyvtár2*] ...]

A *meghajtó* megnevezésének elhagyásakor az utasítás az éppen aktív meghajtóra fog vonatkozni. A **ChDir** utasítás csak az alapértelmezettnek tekintendő könyvtár nevét módosítja, azaz ha az utasításban az alapértelmezettől eltérő meghajtót nevezünk meg, az csak a könyvtárból való keresés vagy beírás időtartamára veszi át az aktív meghajtó szerepét, azt követően ismét az eredeti meghajtó válik aktívvá.

ChDrive

Szintaxis: **ChDrive** *meghajtó*

Az utasítás a *meghajtó* argumentumban adott karakteres kifejezés első karakterével meghatározott meghajtót teszi aktívvá. Ha a meghajtónév üres ("") karakterlánc, továbbra is az éppen érvényes meghajtó marad aktív.

Close

Szintaxis: **Close** [*fájlszámlista*]

Az utasítást fájl vagy fájlok ki- és bemeneti műveletek számára történő lezárására használhatjuk. A lezárandó fájlokat a feltételes *fájlszámlista* argumentumban, a megnyitásukkor az **Open** utasításban használt fájlszámaik egymástól vesszővel elválasztva történő felsorolásával jelöljük ki: [[#]*fájlszám1*] [, [#]*fájlszám2*] ... Ha az utasítást argumentum nélkül használjuk, az **Open** utasítással megnyitott minden egyes aktív fájl

lezárul.

Az **Append** és **Output** hozzáférésre megnyitott fájlok lezárásakor a kimenet utolsó puffertartalma az operációs rendszer pufferébe kerül. A fájl lezárásakor egyben a megnyitásakor használt fájlszám, és a memóriában hozzá tartozó pufferterület is felszabadul. A pufferterület előzőleg lemezre történő mentéséhez a **Reset** utasítás használható.

CommitTrans

Szintaxis:

Ez az argumentum nélküli — csak kompatibilitási okokból megmaradt — utasítás az egyetlen egységként tekintett adatbázisokban végzett módosítások sorozatát, azaz egy tranzakciót zár le. A tranzakciók megnyitása óta az adatbázisokban végzett módosítások érvényesek maradnak, azokat már a csak egy újabb tranzakció keretében lehet megváltoztatni. **Workspace** objektumokkal a tranzakció lezárására inkább a **CommitTrans** metódus használandó.

CompactDatabase

Szintaxis:

Ez a bezárt adatbázis másolását, a töredezettség megszüntetésével történő tömörítését, továbbá verziószámának, összehasonlítási rendjének és dekódolásának módosítását lehetővé tevő utasítás csak kompatibilitási okokból maradt meg. **Workspace** objektumokkal ezekre a feladatokra inkább a **CompactDatabase** metódus használandó.

Const

Szintaxis: [**Public** | **Private**] **Const** *konstansnév* [**As** *típus*] = *kifejezés*

Az utasítással konstansok deklarálhatók, amelyek az adott program számára a továbbiakban változatlanul és a felhasználó számára megváltoztathatatlanul megőrződnek.

- A feltételes **Private** vagy **Public** kulcsszavakkal egy modul deklarációs részében szabhatjuk meg a konstans hatókörét. Az alapértelmezett **Private** csak a deklarációt tartalmazó modulon belül, míg a **Public** az összes modul minden eljárásában érvényes állandót hoz létre. Az eljáráson és az osztálymodulon belüli konstansok mindig az alapértelmezett hatókörűek lesznek, míg a standard modulokban a hatókör a **Public** kulcsszóval módosítható.
- A konstansra a deklaráció után a *konstansnév* argumentumban adott néven lehet hivatkozni.
- A konstans hatókörén belüli értékét a *kifejezés* argumentum szabja meg, amelyben már deklarált szimbolikus konstanst, nem szimbolikus konstanst, aritmetikai

és (az **Is** művelettől eltekintve) logikai műveletet tartalmazó kifejezést, valamint ezek kombinációját is használhatjuk, azonban a *kifejezés* értelemszerűen nem tartalmazhat változókat, továbbá beépített vagy a felhasználó által létrehozott függvényeket sem.

- A deklarált konstans típusát explicit módon a feltételes **As** *típus* záradékkal írhatjuk elő. Ha az explicit meghatározása hiányzik, a konstans típusát a *kifejezés* szabja meg. Az érvényes konstanstípusok a következők: **Byte**, **Boolean**, **Integer**, **Long**, **Currency**, **Single**, **Double**, **Date**, **String** és **Variant**.

Egyetlen utasításban több konstanst is deklarálhatunk egymástól vesszővel elválasztott argumentumblokkokat használva. Ilyen esetben a **Private** vagy **Public** kulcsszavak használata mindegyik konstanshoz azonos hatókört szab meg, de az **As** *típus* záradék a konstansokra egyedi adattípusok deklarálását is megengedi.

A **Sub**, **Function** vagy **Property** eljárásokban deklarált konstansok az adott eljárásra nézve lokálisak, míg az eljárásokon kívül deklaráltak a deklarációt tartalmazó modulra nézve modulszintűek.

Date

Szintaxis: **Date** = *dátum*

Ezzel az utasítással a rendszerben a program futása alatt érvényes dátum állítható be. MS-DOS gépekben a *dátum* argumentum 1980. január 1. és 2099. december 31, a Windows NT gépekben pedig 1980. január 1. és 2079. december 31 között lehet, egyébként hibajelzést kapunk. Az utasítás az MS-DOS 3.3-s előtti verziókban nem módosítja a számítógép CMOS RAM-jában tárolt dátumot, vagyis a beállított dátum csak a számítógép kikapcsolásáig marad érvényes. Ilyen rendszerekben a dátum tartós módosítása csak a rendszer dokumentációjában leírt módon lehetséges.

Declare

Szintaxis 1: [**Private** | **Public**] **Declare Sub** *név* **Lib** "*könyvtárnév*" [**Alias** "*hivatkozási_név*"] [([*DLLargumentumlista*])]

Szintaxis 2: [**Private** | **Public**] **Declare Function** *név* **Lib** "*könyvtárnév*" [**Alias** "*hivatkozási_név*"] [([*DLLargumentumlista*])] [**As** *függvénytípus*]

Ezzel a modulszinten használható, meglehetősen összetett utasítással dinamikusan csatolt eljárás és függvénytárakban (*Dynamic Link Library*, DLL) található külső eljárások számára történő hivatkozások deklarálhatók. Az utasítás két, alakilag csak kismértékben eltérő szintaxissal használatos: az első szintaxisában a **Sub** kulcsszó azt jelzi, hogy a külső eljárás nem ad vissza eredményt, míg a másodikban szereplő **Function** azt, hogy a külső eljárás függvényként működik, tehát valamilyen eredményt ad vissza, és

így kifejezésekben is alkalmazható.

- Az utasításban feltételesen használható **Private** vagy **Public** záradékkal a DLL hatóköre írható elő: az alapértelmezett **Private** deklarációval az adott DLL használata csak a deklarációt tartalmazó modulra korlátozódik, míg a **Public** deklarációval a DLL minden modul összes eljárásai számára is elérhető lesz.
- Az utasítás *globális_név* argumentuma egy érvényes **Sub** vagy a **Function** eljárás neve.
- Az utasítás következő részében a **Lib** "*könyvtárnév*" azt adja meg, hogy a deklarálandó eljárás vagy függvény egy, a *könyvtárnév* argumentumban adott karakteres kifejezésnek megfelelő nevű DLL-ben található.
- Ha a DLL-ben a deklarált eljárás nem a *név*, hanem a *hivatkozási_név* argumentumban adott név alatt található, ezt a feltételes **Alias** "*hivatkozási_név*" jelzi. Az ilyen esetében a külső eljárás neve akár egy VBA kulcsszó, egy már lefoglalt globális változó, állandó vagy eljárás neve, esetleg más, a DLL névadási konvenciói szerint tiltott név vagy tiltott karaktert tartalmazó név is lehet. A nem # karakterrel kezdődő *hivatkozási_név* a DLL belépési pontjának nevét, a # karakterrel kezdődő *hivatkozási_név* pedig az eljárás DLL-en belüli belépési pontjának sorszámát kell hogy megadja.
- A deklaráció következő része a *DLLargumentumlista*, amelyben a DLL számára átadandó változók és típusaik egymástól vesszővel elválasztott listáját tartalmazza az [**Optional**] [**ByVal** | **ByRef**] [**ParamArray**] *változónév* [( )] [**As** *típus*] alakú általánosított felépítésben.
  - A feltételes **Optional** azt jelzi, hogy az ezt követő argumentum feltételes, azaz az adott eljárás meghívásakor nem kötelező megadni. Ha egy argumentum feltételes, a *DLLargumentumlista* ezt követő további argumentumainak is feltételeseknek, azaz az **Optional** kulcsszóval deklaráltaknak és **Variant** típusúaknak kell lenniük. Az **Optional** és a **ParamArray** kulcsszavak egy DLL argumentumon belül nem használhatók.
  - A **ByVal** vagy **ByRef** feltételes kulcsszó a változóértékek külső eljárás számára történő átadási módját határozza meg. A **ByVal** esetén a numerikus adattípusok éppen aktuális értéke átadás előtt a **Declare** által meghatározott adattípussá konvertálódik, míg a **String** típusú változó esetében a nullával lezárt karakterlánc címe adódik át az eljárásnak. A **ByVal** használata nélkül a **String** típusú változó karaktersorozat-leírásként kerül át a hívott DLL eljárásba. A **ByVal** ellentéte a **ByRef**, ami hatására az azt követő argumentum referenciával (címmel) adódik át a meghívott eljárásnak.
  - Az ugyancsak feltételes **ParamArray** csak az argumentumlista utolsó eleme előtt állhat, jelezve, hogy az adott argumentum egy feltételes **Variant** elemekből álló **Optional** tömb. Ilyen módon tetszőleges számú és típusú további argumentumot adhatunk át egy DLL eljárásnak, természetesen csak

akkor, ha az felkészült ezek fogadására. A **ParamArray** kulcsszóval együtt a **ByVal**, **ByRef** és **Optional** kulcsszavak egyike sem szerepelhet.

- A lista egyetlen kötelező eleme a külső eljárásnak átadandó, a változónevek megadására vonatkozó konvencióknak megfelelő *változónév* argumentum. Ha egy változó adattömb, azt a tömb indextartományának megadása nélkül, üres zárójelpárral kell jelezni.
- Az eljárásnak átadandó változó típusát a feltételes **As** *típus* adja meg. A használható adattípus **Boolean**, **Byte**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, kizárólag változó hosszúságú **String**, **Variant**, felhasználói által definiált vagy egy objektumtípus lehet. Az állandó hosszúságú karakterláncok az eljárásnak történő átadás előtt változó hosszúságúakra konvertálódnak. Ha azt kívánjuk, hogy az eljárásnak bármilyen típusú változó átadható legyen, az **As** *típus* helyett az **As Any** típusjelölést is használhatjuk.

- A **Function** eljárások esetén minden egyes meghívásakor a változók száma és — hacsak nem az **As Any** típusjelölést alkalmaztuk — típusa is ellenőrzésre kerül. A használt változótípusok egyben megszabják a visszakapott adattípust, azonban az argumentumlistát követően elhelyezett, feltételes **As** *függvénytípus* használatával ezt mi is előírhatjuk.

A **Declare** utasítást üres zárójelpárral deklarálva azt jelöljük, hogy az adott **Sub** vagy **Function** eljárás számára nem kell argumentumokat átadni. Az így deklarált eljárás argumentummal történő használata hibát okoz.

Ha a meghívandó eljárás nulla értékű — nem üres ("") — karakterláncot vár, a **vbNullString** szimbolikus konstans is használható.

Deftípus

Szintaxis: **Def***típus betűtartomány* [, *betűtartomány1*] ...

Ezzel a modulszintű utasítással adott tartományba eső betűvel kezdődő nevű változók, eljárásoknak átadandó argumentumok, valamint a **Function** és **Property Get** eljárások által visszaadott értékek alapértelmezetten **Variant** típusa helyett egy modulon belül a következő táblázat szerinti típusok beállítása lehetséges:

Ha a **Def***típus*:	Az adattípus:
DefBool	**Boolean**
DefByte	**Byte**
DefCur	**Currency**
DefDate	**Date**
DefDbl	**Double**
DefInt	**Integer**

DefLng	**Long**
DefObj	**Object**
DefSng	**Single**
DefStr	**String**
DefVar	**Variant**

Az egyes adattípusdeklaráló utasítások argumentumaiban szereplő, *betű1*[*-betű2*] alakban írható *betűtartomány* az adott típussal érvényes nevek kezdőbetű-tartományának megadását teszi lehetővé: így például a `DefInt A-Z` utasítás hatására minden — így a 128-255 között található karakterekkel kezdődő — változó az utasítást tartalmazó modulon belül **Integer** típusú lesz. (A *betűtartomány* megadásában a kis- és nagybetűk egyenértékűek.) A deklarációt követően a változónevek egy résztartományának adattípusa csak az "erősebb" **Dim** *változó* **As** *típus* deklarációval módosítható, a **Def***típus* utasítással megkísérelt módosítás hibát okoz.

A **Def***típus* utasítás a felhasználó által definiált típusok elemeire hatástalan, mivel azok explicit módon deklaráltak.

DeleteSetting

Szintaxis: **DeleteSetting(** ***appname***, ***section*** [, ***key***])

Az utasítással a Windows regisztrációs bejegyzései közül egy szakaszt vagy kulcsot lehet törölni.

- A névvel hivatkozott ***appname*** argumentum karakteres kifejezésében kell megadni annak az alkalmazásnak vagy projektnek a nevét, amelyhez a szakasz vagy kulcs tartozik.
- A ***section*** argumentum karakteres kifejezés a törlendő szakasz vagy a törlendő kulcsot tartalmazó szakasz nevét tartalmazza.
- A feltételes ***key*** argumentum karakteres kifejezésében a törlendő kulcsnak kell szerepelnie.

Ha a szükséges, névvel hivatkozott argumentumok adottak, és az általuk kijelölt szakasz és kulcs létezik, az utasítás törli azt, egyébként hatástalan marad.

Dim

Szintaxis: **Dim** *változónév* [([*indextartomány*])] [**As** [**New**] *típus*] [, ...]

Az utasítással változók modulszintű, a **Sub** vagy **Function** eljáráson belül használva pedig eljárásszintű deklarálása, valamint és a hozzájuk tartozó tárolóterület kiosztása, allokálása lehetséges. Egy modulon belül történő használatát követően a deklarált változó minden, az adott modulon belüli eljárás számára is elérhető lesz, míg az eljárásszinten történt deklarálást követően a változó csak az adott eljáráson belül használható.

A **Dim** argumentumai a változóneveket és azok típusát, továbbá a dimenzionált tömbök indextartományait tartalmazzák.

- A *változónév* a dimenzionálni kívánt változónak a szokásos névadási konvenciónak megfelelő nevét tartalmazza.
- A feltételes *indextartomány* argumentum a maximálisan 60 dimenziós tömbváltozók listáját tartalmazza [*alsóindex* **To**] *felsőindex* alakban, a lista elemeit egymástól vesszővel elválasztva. Ha a feltételes [*alsóindex* **To**] nincs megadva, az *alsóindex* alapértelmezett értéke az **Option Base** beállításától függően lesz 0 vagy 1.
- A használható változók típusai a feltételes **As** [**New**] *típus* részben deklarálhatók, ahol **Boolean**, **Byte**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, **Single**, **String** (változó hosszúságú), **String** * *hosszúság* alakú (rögzített hosszúságú) karakterlánc, **Variant**, felhasználó által definiált típus vagy objektum típus közül adható meg a kívánt adattípus. Minden egyes, akár azonos típusú változóhoz is külön-külön **As** *típus* meghatározás tartozik. Az adat- vagy objektívtípus nélkül deklarált változók — hacsak egy **Def***típus* utasítással nincs más típus előírva — alapértelmezetten **Variant** típusúak lesznek.
    - A típusdeklarációban a feltételes **New** kulcsszóval azt jelezhetjük, hogy a deklarált objektumváltozó egy Visual Basic objektum egy új példánya vagy egy külsőleg létrehozható OLE Automatizmus objektum. Ha objektumváltozó deklarációban nem használt a **New**, az objektumnak nem létezik példánya. A **New** kulcsszó nem használható belső adattípusú változó vagy függő OLE Automatizmus objektumok példányainak deklarálására. Az objektumváltozó csak létező, használat előtt a **Set** utasítással értéket kapott objektumra hivatkozhat, egyébként a deklarált objektumváltozó értéke **Nothing**.

Üres változótömbbel adott **Dim** utasítással a Visual Basic előző verzióiban ismert dinamikus, vagyis a felhasználás során meghatározható nagyságú tömbök is deklarálhatók. A deklarálás után a dinamikus tömbök dimenziójának és elemeinek száma eljárásszinten a **ReDim** utasítással definiálhatók. A **Dim**, **Private** és **Public** utasítással explicit módon létrehozott tömbváltozó újradimenzionálása hibát okoz.

A változók inicializálásakor a **Dim** utasítás minden numerikus változót és tömböt nulláz, a rögzített hosszúságú karakterláncot nullákkal, a változó hosszúságút üres ("") karakterláncokkal, a **Variant** típusúakat pedig **Empty** értékekkel tölti fel. A felhasználó által definiált változótömbök az egyedi változókkal azonos módon kapnak kiinduló értéket.

Mint minden eljárásszintű deklarációt, a **Dim** utasítást is az eljárás elején szokás elhelyezni.

Do...Loop

Szintaxis 1:

Do [{**While** | **Until**} *feltétel*]
[*utasításcsoport1*]
[**Exit Do**]
[*utasításcsoport2*]
Loop

Szintaxis 2:

Do
[*utasításcsoport1*]
[**Exit Do**]
[*utasításcsoport2*]
Loop [{**While** | **Until**} *feltétel*]

Az utasítás egy utasítás vagy utasításcsoport egy feltételtől függő, előre nem ismert számú, ismétlődő végrehajtását eredményezi. A *feltétel* argumentum valamilyen numerikus vagy karakteres kifejezések közötti logikai reláció, míg a feltételes *utasításcsoport1* és *utasításcsoport2* argumentumok egy vagy több utasítást tartalmazhat, amelyek végrehajtása mindaddig megismétlődik, ameddig a *feltétel* kiértékelése **True** eredményt ad.

Az első szintaxis **While** kulcsszót tartalmazó változatában a *feltétel* vizsgálatának **True** eredménye esetén az *utasításcsoport1* végrehajtásra kerül. Ez lehet egy **If...Then** felépítésű vizsgálat, amely pl. a ciklus adott számú lefutása után eredményezi a ciklusból a **Loop** utáni sorra történő kilépést, míg a ciklus hasznos utasításai az *utasításcsoport2* csoportban vannak. A ciklusból a **Do...Loop** belsejében tetszőleges számban elhelyezhető, feltételes **Exit Do** utasítás ad kilépési lehetőséget.

Ezzel közel azonos a második szintaxis működése, mindössze az az eltérés, hogy az *utasításcsoport1* legalább egyszer mindenképpen végrehajtódik, hiszen a *feltétel* vizsgálata csak ezt követően történik meg. Ha most a feltétel **True**, a ciklus visszatér az *utasításcsoport1* végrehajtására, ellenkező esetben a program futása a **Loop While** *feltétel* utáni sorban folytatódik.

Hasonló felépítésűek és működésűek az **Until** kulcsszót tartalmazó alakok, de itt az *utasításcsoport1* végrehajtása nem a feltétel **True**, hanem **False** eredményének fennállásáig ismétlődik.

A **Do...Loop** ciklusok tetszőleges számban egymásba ágyazhatók, és egy belső ciklusból az **Exit Do** utasítás az eggyel magasabb szintű ciklus számára adja át a vezérlést.

End

Szintaxis: **End [Function | If | Select | Sub | Type | Property | With]**

Az **End** utasítás lezárja egy eljárás vagy blokk futását, töröli az összes modulszintű változót és minden modul statikus helyi változóját.

Az utasítás feltételes kulcsszavakkal képezhető különböző alakjait és azok hatását a következő táblázat foglalja össze:

Alak:	Hatása:
End	Lezárja a program futását. A programban bárhol elhelyezhető az **Open** utasítással megnyitott fájlok bezárásához és a változók értékének törléséhez.
End Function	Lezár egy **Function** eljárást. Egy új **Function** eljárás megkezdésekor automatikusan a programvázhoz adódik.
End If	Lezár egy **If...Then...Else** utasítást.
End Select	Lezár egy **Select Case** utasítást.
End Sub	Lezár egy **Sub** eljárást. Egy új **Sub** eljárás megkezdésekor automatikusan a programvázhoz adódik.
End Type	Lezár egy felhasználói típusdeklarációra használt **Type** utasítást.
End Property	Lezárja a **Property Let**, **Property Get** és a **Property Set** eljárásokat.
End With	Lezár egy **With** utasítást.

Mivel az **End** utasítás törli a változókat, minden olyan esetben, ha a változók értékét meg kell őrizni, a **Stop** utasítást kell használni.

Erase

Szintaxis: **Erase** *tömblista*

Az utasítás a *tömblista* argumentumban felsorolt, vesszővel elválasztott nevű tömbök közül az állandó méretűeket újrainicializálja, a dinamikus tömbök tárolóterületét pedig felszabadítja.

Állandó méretű tömbnél az **Erase** utasítás:

- a numerikus tömböket nullákkal,
- a változó hosszúságú karakteres tömböket üres ("") karakterekkel,
- az állandó hosszúságú karakteres tömböket nullákkal,
- a **Variant** tömböket **Empty** értékekkel,
- a felhasználói típusú tömböt az elemek típusának megfelelően,
- az objektumok tömbjét pedig **Nothing** elemekkel tölti fel.

A dinamikus tömb tömbváltozóinak dimenzióit a tömb újbóli használata előtt a **ReDim** utasítással újra kell deklarálni.

Error

Szintaxis:

A programfutási hiba bekövetkezését szimuláló **Error** utasítás a Visual Basic 4.0-s verziójában csak kompatibilitási okból maradt meg, helyette — különösen OLE Automatizmus objektumok létrehozásakor — az **Err** objektumra alkalmazott **Raise** metódussal lehet futásidejű hibákat előállítani.

Exit

Szintaxis: **Exit {Do | For | Function | Sub | Property}**

Az **Exit** utasítással egy eljárásciklusból — az eljárás lezárására használatos **End** utasítástól eltérő módon — lehet kilépni.

Az utasítás kulcsszavakkal képezhető különböző alakjait és azok hatását a következő táblázat foglalja össze:

Alak:	Hatása:
Exit Do	Kilépés egy **Do...Loop** ciklusból. A vezérlés a **Loop** kulcsszó utáni sorra kerül. Egymásbaágyazott ciklusoknál a vezérlés az **Exit Do** utasítást tartalmazó ciklusból az eggyel magasabb szintű ciklus számára adódik át.
Exit For	Kilépési egy **For...Next** vagy **For Each...Next** ciklusból. A vezérlés a **Next** kulcsszó utáni sorra kerül. Egymásbaágyazott ciklusoknál a vezérlés az **Exit For** utasítást tartalmazó ciklusból az eggyel magasabb szintű ciklus számára adódik át.
Exit Function	Azonnali kilépés egy **Function** eljárásból. A program futása a **Function** eljárást meghívó utasítást követő utasítás végrehajtásával folytatódik.
Exit Property	Azonnali kilépés egy **Property** eljárásból. A program futása a **Property** eljárást meghívó utasítást követő utasítás végrehajtásával folytatódik.
Exit Sub	Azonnali kilépés egy **Sub** eljárásból. A program futása a **Sub** eljárást meghívó utasítást követő utasítás végrehajtásával folytatódik.

A kizárólag eljárásban vagy programciklusban használható **Exit** utasítás nem az adott eljárás vagy programciklus végét jelzi.

FileCopy

Szintaxis: **FileCopy** ***source, destination***

Az utasítással egy, a névvel hivatkozott ***source*** argumentum karakteres kifejezésé-

ben adott nevű fájlról a ***destination*** argumentum karakteres kifejezésében adott névvel lehet másolatot készíteni. Az argumentumok a meghajtó és elérési út megjelölését is tartalmazhatják. A nyitott fájlok másolásának kísérlete hibát okoz.

For Each...Next

Szintaxis:

For Each *elem* **In** *csoport*
[*utasításcsoport1*]
[**Exit For**]
[*utasításcsoport2*]
Next [*elem*]

A **For Each...Next** utasítás ciklikusan egy utasítást vagy utasítások csoportját hajtja végre egy objektumgyűjtemény vagy tömb minden egyes elemén.

- A szintaxisban az *elem* argumentumban adott változó egy objektumgyűjtemény vagy tömb azon elemeit jelöli ki, amelyen adott utasításokat akarunk végrehajtatni. Ha az *elem* egy adott változó egy tömb elemére vonatkozik, akkor az csak **Variant** típusú lehet, ha egy halmaz elemére, akkor **Variant**, eredeti **Object** vagy OLE Automatizmus objektum típusú.
- A szintaxisban az **In** *csoport* argumentum jelöli ki azt a — legalább egy elemet tartalmazó — objektumgyűjteményt vagy tömböt, amelynek elemein az utasítások végrehajtása történik. A tömb nem lehet felhasználó által definiált típus.
- A szintaxisban az *utasításcsoport1* és *utasításcsoport2* argumentum a végrehajtandó utasításokat tartalmazza.
- A cikluson belül tetszőleges számban elhelyezhető **Exit For** a ciklusból történő kiugrást, és a program futásának a **Next** utáni utasítással történő folytatását teszi lehetővé. Az **Exit For** utasítást általában egy feltételes utasítás, például **If...Then** egyik ágaként szokták használni.

A ciklus minden lefutása kezdetén ellenőrződik, hogy az éppen érvényes *elem* ciklusváltozó a *csoport* eleme-e. Ha igen, végrehajtódik a ciklus belsejében szereplő összes utasítás, majd a **Next** hatására az *elem* argumentumba a csoport következő eleme kerül. A **Next** után nem kötelező az *elem* argumentumot megadni, azt a program automatikusan értelmezi.

Az utasítás végrehajtása normálisan akkor fejeződik be, amikor elfogynak a *csoport* elemei. Az utasítás végrehajtása után a programfutás a **Next** utáni utasítással folytatódik. Ha a *csoport* egyetlen elemet sem tartalmaz, a **For Each...Next** ciklus egyszer sem fut le.

A **For Each...Next** ciklusok tetszőlegesen egymásba ágyazhatók, azonban mindegyik ciklusban csak egyedi *elem* argumentumot szabad használni.

For...Next

Szintaxis:

For *ciklusváltozó* = *első* **To** *utolsó* [**Step** *lépésköz*]
[*utasításcsoport1*]
[**Exit For**]
[*utasításcsoport2*]
Next [*ciklusváltozó*]

A **For...Next** utasítás egy utasítás vagy utasításcsoport — a **For** sorban beállított argumentumoknak megfelelő — számú végrehajtását eredményezi.

- A szintaxisban a *ciklusváltozó* argumentum egy numerikus változó, amelynek kiinduló értékét az *első*, végső értékét pedig az *utolsó* értéke adja meg. A *ciklusváltozó* nem lehet egy tömb eleme vagy felhasználó által definiált típusú.
- A *ciklusváltozó* léptetése az *első* és az *utolsó* értéke között alapértelmezetten egyesével történik. Ha ettől el akarunk térni, az új — akár pozitív, akár negatív léptetési értéket a feltételes **Step** *lépés* argumentumban kell megadni. Természetesen a *lépés* előjelének összhangban kell állnia az *első* és az *utolsó* értékei által meghatározott iránnyal. Noha nem kötelező, általában az *első*, *utolsó* és *lépés* argumentumok **Integer** típusúak. A *ciklusváltozó* értékének a cikluson belül történő megváltoztatása előre nem látható hatásokkal járhat.
- A szintaxisban az *utasításcsoport1* és *utasításcsoport2* argumentum a végrehajtandó utasításokat tartalmazza.
- A cikluson belül tetszőleges számban elhelyezhető **Exit For** a ciklusból történő kiugrást, és a program futásának a **Next** utáni utasítással történő folytatását teszi lehetővé. Az **Exit For** utasítást általában egy feltételes utasítás, például **If...Then** egyik ágaként szokták használni.

Az utasítás legegyszerűbb alakja esetén a *ciklusváltozó* felveszi az *első* értéket, majd egy vizsgálat következik, amely ellenőrzi, hogy a *ciklusváltozó* értéke nagyobb-e az *utolsó* értéknél. Ha igen, a program a **Next** sor utáni sorba lép, ellenkező esetben (ha van) végrehajtja a ciklusmagban szereplő utasításokat, majd eggyel megnöveli a *ciklusváltozó* értékét, és visszatér a **For...Next** ciklus elejére. A ciklus utolsó végrehajtása akkor történik, ha az *első* és az *utolsó* értéke megegyezik, majd a programfutás a **Next** utáni utasítással folytatódik. A **Next** után nem szükséges a ciklusváltozó feltüntetése, azt a program automatikusan értelmezi.

Az utasítást üres ciklusmaggal is használhatjuk a programba várakozás vagy durva időzítés beiktatása céljából.

A **For...Next** cikluson belül másik, eltérő ciklusváltozójú **For...Next** is használható, azaz a ciklusok tetszés szerinti mélységig egymásba ágyazhatók.

FreeLocks

Szintaxis:

Ez az adatfeldolgozás félbeszakítását, a rekordoldalak zároltságának feloldását lehetővé tevő és a helyi **Dynaset** objektum összes adatát a többfelhasználós környezetben aktuálissá tevő utasítás csak kompatibilitási okokból maradt meg. Helyette az **Idle** metódus **dbFreeLocks** szimbolikus konstanssal történő használata javasolt.

Function

Szintaxis:

[**Private** | **Public**] [**Static**] **Function** *eljárásnév* [(*argumentumlista*)] [**As** *típus*]
[*utasításcsoport1*]
[*eljárásnév* = *kifejezés*]
[**Exit Function**]
[*utasításcsoport2*]
[*eljárásnév* = *kifejezés*]
End Function

Ez az utasítás a Visual Basic beépített függvényeihez hasonlóan egy kifejezés jobb oldalán használható **Function** eljárás nevét, argumentumait és programkódját deklarálja.

- Az utasításban feltételesen használható **Private** vagy **Public** záradékkal a **Function** eljárás hatóköre írható elő: az alapértelmezett **Private** deklarációval az eljárás használata csak a deklarációt tartalmazó modulra korlátozódik, míg a **Public** deklarációval minden modul összes eljárásai számára is elérhető lesz.
- Az ugyancsak feltételes **Static** záradék azt szabja meg, hogy a **Function** eljárás helyi változóinak értékei a hívások között is megtartsák értéküket. Az eljárásban használt nem helyi változókra a **Static** hatástalan.
- A **Function** eljárásnak a változónevekre érvényes konvencióknak megfelelően képzett nevét az *eljárásnév* argumentumban kell megadni.
- Az eljárás nevét az eljárásban használt, annak meghívásakor átadandó változó a következő, [**Optional**] [**ByVal** | **ByRef**] [**ParamArray**] *változónév* [( )] [**As** *típus*] általánosított alakú, több változó esetén azok vesszővel elválasztott, zárójelbe tett listája, az *argumentumlista* követi.
    - A feltételes **Optional** azt jelzi, hogy az ezt követő argumentum feltételes, azaz az adott eljárás meghívásakor nem kötelező megadni. Ha egy argumentum feltételes, az *argumentumlista* ezt követő további argumentumainak is feltételeseknek, azaz az **Optional** kulcsszóval deklaráltaknak és **Variant** típusúaknak kell lenniük. Az **Optional** és a **ParamArray** kulcsszavak egy argumentumon belül nem használhatók.

- A **ByVal** vagy **ByRef** feltételes kulcsszó a változóértékek külső eljárás számára történő átadási módját határozza meg. A **ByVal** esetén a numerikus adattípusok éppen aktuális értéke átadás előtt a **Declare** által meghatározott adattípussá konvertálódik, míg a **String** típusú változó esetében a nullával lezárt karakterlánc címe adódik át az eljárásnak. A **ByVal** használata nélkül a **String** típusú változó karaktersorozat-leírásként kerül át a hívott **Function** eljárásba. A **ByVal** ellentéte a **ByRef**, ami hatására az azt követő argumentum referenciával (címmel) adódik át a meghívott eljárásnak.
- Az ugyancsak feltételes **ParamArray** csak az argumentumlista utolsó eleme előtt állhat, jelezve, hogy az adott argumentum egy **Variant** elemekből álló **Optional** tömb. Ilyen módon tetszőleges számú és típusú további argumentumot adhatunk át egy eljárásnak, természetesen csak akkor, ha az felkészült ezek fogatására. A **ParamArray** kulcsszóval együtt a **ByVal**, **ByRef** és **Optional** kulcsszavak egyike sem szerepelhet.
- A lista egyetlen kötelező eleme a külső eljárásnak átadandó, a változónevek megadására vonatkozó konvencióknak megfelelő *változónév* argumentum. Az eljárásban nem expliciten deklarált változók használata esetén névütközési problémák következhetnek be, ezért a programkódban célszerű az **Option Explicit** utasítással kötelező deklarálást előírni.
- Az eljárásnak átadandó változó típusát a feltételes **As** *típus* adja meg. A használható adattípus **Boolean**, **Byte**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, kizárólag változó hosszúságú **String**, **Variant**, felhasználói által definiált vagy egy objektumtípus lehet.

- A **Function** eljárás eredményének típusát általában az argumentumokban használt változók típusa határozza meg, azonban az argumentumlista végén elhelyezett feltételes **As** *típus* deklarációval explicit típuselőírásra is lehetőség van. A *típus* a következők valamelyike lehet: **Byte**, **Boolean**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, **Single**, **String** (csak változó hosszúságú karakterlánc), **Variant**, vagy bármely felhasználói típus. Semmilyen típusú tömböt nem kaphatunk vissza, azonban tömböt tartalmazó **Variant** értéket igen.
- A **Function** eljárásban végrehajtandó utasításokat a feltételes *utasításcsoport1* és *utasításcsoport2* tartalmazza.
- A **Function** eljárás által visszaadandó értéket az eljárásban tetszőleges helyen és számban elhelyezhető *eljárásnév* = *kifejezés* határozza meg. Ha nem történik ilyen eredményátadás, az *eljárásnév*, azaz az eljárás eredménye numerikus függvénynél 0, karakteres függvénynél üres ("") karakterlánc, **Variant** függvény esetén pedig **Empty** érték lesz. Ha az objektumhivatkozást visszaadó függvényben a **Set** utasítással nem történik objektumhivatkozás beállítása az *eljárásnév* argumentumba, a **Function** eljárás **Nothing** eredményt ad vissza.

- A tetszőleges számban elhelyezhető **Exit Function** kulcsszóval lehet a **Function** eljárásból kilépni. Ekkor a programfutás az eljárást meghívó utasítást követő utasítás végrehajtásával folytatódik.
- A **Function** eljárást az **End Function** zárja le.

A **Function** eljárást az *eljárásnév* és az azt követő (*argumentumlista*), azaz a zárójelben felsorolt argumentumok megadásával lehet meghívni.

A **Function** elsőként az *utasításcsoport1* utasítást vagy utasításcsoportot (általában valamilyen vizsgálatot) hajtja végre, majd a visszatérési érték beállítása után vagy a függvényt megvalósító *utasításcsoport2* utasítás vagy utasításcsoport végrehajtását kezdi meg, vagy az **Exit Function** utasításon keresztül azonnal a **Function** eljárást hívó utasítás utáni utasításra lép.

A függvényt megvalósító utasításblokkban az adott függvény is meghívható, azaz megengedett a függvények rekurzív hívása, de gondoskodni kell arról, hogy a rekurzió ne legyen végtelen. A rekurzívan használandó függvények deklarálásakor ne használjuk a **Static** kulcsszót.

A Function eljárás használatát kerülni kell az olyan numerikus kifejezésekben, amely ugyanazon változókat használja, mint a **Function**.

Function eljárást nem definiálhatunk másik **Function**, **Sub** vagy **Property** eljárásokon belül.

Get

Szintaxis: **Get** [#] *fájlszám*, [*rekordszám*], *változónév*

Az utasítás egy nyitott lemezfájlból egy adott rekordot olvas be.

- A *fájlszám* argumentumaként az **Open** utasításban a fájl megnyitására használt számot használva, az utasítás egy nyitott lemezfájlból egy adott rekordot olvas be.
- A beolvasandó rekordot **Random** fájl esetén a rekordszám, **Binary** fájl esetén a byte-szám értékét tartalmazó, feltételes *rekordszám* argumentum tartalmazza. Ha a *rekordszám* hiányzik, a következő beolvasandó rekord vagy byte az utolsó **Get** vagy **Put** utasítás, illetve az utolsó **Seek** utasítással kijelölt utáni lesz.
- A beolvasott rekord a *változónév* argumentumban adott nevű változóba kerül.

A fájlból történő olvasás során a fájl megnyitási módjától függően néhány szempontot és szabályt kell figyelembe venni.

- **Random** móddal megnyitott fájloknál:
  - Ha az olvasandó adatok hossza kevesebb, mint az **Open** utasítás **Len** értékeként megadott rekordhossz, a két rekord között fennmaradó helyre a puffer pillanatnyi tartalmából kerülnek később már azonosíthatatlan meny-

nyiségű, ezért a hasznos adattól szétválaszthatatlan értékek, ezért célszerű a rekordhosszat és az írandó adatmennyiséget azonosra választani.

- Ha az olvasandó változó egy változó hosszúságú karakterlánc, a **Get** egy 2 byte-os hosszadat alapján tölti be a változót, tehát az **Open** utasításban **Len** értékként megadott rekordhosszban a karakterlánc hosszánál legalább 2 byte-tal nagyobb értéket kell megadni.
- Ha az olvasandó változó numerikus **Variant** adat, a **Get** elsőként egy 2 byte-os **Variant VarType** típusazonosítót, majd a típustól függő adat bytejait olvassa be. Az **Open** utasításban **Len** értékként megadott rekordhosszat a változó tárolásához szükséges hosszánál legalább 2 byte-tal nagyobbra kell megadni.
- A **Get** utasítással **Variant** tömböt lehet olvasni, azonban tömböt tartalmazó **Variant** adatot nem.
- Ha az olvasandó változó karakteres **Variant** adat (**VarType** 8), a **Get** elsőként egy 2 byte-os típusazonosítót és egy 2 byte-os hosszadatot olvas be, tehát **Open** utasításban a karakterlánc aktuális hosszánál legalább 4 byte-tal nagyobb **Len** értékként megadott rekordhosszat kell megadni.
- Ha az olvasandó változó dinamikus tömbben található, a **Get** elsőként egy 2 byte + 8**dimenziószám* hosszúságú azonosítót olvas be, tehát **Open** utasításban **Len** értékként megadott rekordhosszat a változók tárolásához szükséges összhossznál legalább ennyivel nagyobbra kell megadni.
- Ha az olvasandó változó rögzített méretű tömbben vagy — a **Variant** és a változó hosszúságú adatokat kivéve — bármely más típusú változóban található, a **Get** nem olvas be azonosítót az adatok elé, tehát **Open** utasításban **Len** értékként megadott rekordhossz a változó tárolásához szükséges hosszal lehet azonos.
- A felhasználó által definiált adattípusok elemeit a **Get** egyedi adatokként, a közöttük lévő helykitöltő elemek nélkül olvassa be. A **Put** utasítással fájlba írt felhasználó által definiált adattípusú dinamikus tömb előtt egy 2 byte + 8**dimenziószám* hosszúságú azonosítót kell beolvasni, tehát **Open** utasításban **Len** értékként megadott rekordhosszat az összes tömböt, azok típus- és hosszadatait tartalmazó egyedi elemek tárolásához szükséges összhossznál legalább ennyivel nagyobbra kell megadni.
- A **Get** utasítással nem lehet objektumokat fájlból olvasni.

- A **Binary** módban megnyitott fájloknál a **Random** módra leírtak az alábbi eltérésekkel érvényesek:
  - Az **Open** utasításban a **Len** értékének nincs hatása. Mivel a **Get** folyamatosan olvassa az adatokat, a rekordok között nincsenek kitöltő elemek.
  - A felhasználó által definiált adattípuson belüli adatok kivételével a **Get** utasítás csak az adatokat olvassa be, a típus- és hosszadatokat nem.

- A **Get** a nem felhasználó által definiált típus elemeként használt változó hosszúságú karakterláncokat a 2-byte-os hosszadat nélkül olvassa be.

GoSub...Return

Szintaxis:

GoSub {*címke* | *sor*}
...
{*címke* | *sor*}
...
Return

A **GoSub...Return** feltétel nélküli vezérlésátadó utasításszerkezetet egy eljáráson belül bárhol használhatjuk valamely szubrutin meghívására, és az abból történő visszatérésre. A *címke* vagy *sor* argumentumban a szubrutint tartalmazó utasításblokk első sorának azonosítására szolgál:

- A *címke* a szubrutin kezdeténél elhelyezett egyedi azonosító karaktersorozat.
- A *sor* a szubrutin kezdeténél elhelyezett, 65529-nél kisebb sorszám.

A **GoSub** és a hozzá tartozó **Return** csak azonos eljárásban használható, használatukkal tilos egy **Sub** eljárásba be- vagy onnan kilépni.

Az utasítás több **Return** utasítást is tartalmazhat, azonban a programfutást már az első **Return** a **GoSub** utáni utasításra adja vissza.

GoTo

Szintaxis: **GoTo** {*címke* | *sor*}

A **GoTo** feltétel nélküli vezérlésátadó utasítás a program végrehajtását az eljáráson belül található, az argumentumban megadott címkéjű vagy sorszámú sorra adja át. A *címke* vagy *sor* argumentumban a szubrutint tartalmazó utasításblokk első sorának azonosítására szolgál:

- A *címke* a szubrutin kezdeténél elhelyezett egyedi azonosító karaktersorozat.
- A *sor* a szubrutin kezdeténél elhelyezett, 65529-nél kisebb sorszám.

A túl sok **GoTo** utasítás áttekinthetetlenné és nehezen áttekinthetővé teszi a program követését, ezért használatát kerülni kell. Minden **GoTo** utasítást tartalmazó programrész átírható a strukturált programozást lehetővé tevő **Do...Loop**, **For...Next**, **If...Then...Else** vagy **Select Case** utasítások használatával.

If...Then...Else

Szintaxis 1:

If *feltétel* **Then** *utasításcsoport1* [**Else** *utasításcsoport2*]

Szintaxis 2:

If *feltétel1* **Then**
    [*utasításcsoport1*]
[**ElseIf** *feltétel2* **Then**
    [*utasításcsoport2*]]
...
[**Else**
    [*utasításcsoport n*]]
End If

Ennek a feltételes utasításnak egyszerűbb, a Szintaxis 1 szerinti egysoros alakja elsőként megvizsgálja a *feltétel* argumentumban szereplő feltételt. és ha a vizsgálat eredménye **True**, a vezérlés átadódik az *utasításcsoport1*, ellenkező esetben pedig a feltételes **ElseIf** *utasításcsoport2* utasítás vagy utasításcsoport végrehajtására.

- Az **If** *feltétel* **Then** argumentum általában valamilyen numerikus vagy karakteres kifejezések közötti egyszerű vagy logikai kulcsszavakkal összetetté alakított reláció, amely kiértékelése **True** vagy **False** eredmény adhat. Ha a *feltétel* **Null**, az utasítás azt **False** értékűnek tekinti. A *feltétel* **IfTypeOf** *objektumnév* **Is** *objektumtípus* alakba írt kifejezés is lehet, ahol az *objektumnév* bármely objektumhivatkozás, az *objektumtípus* pedig bármely érvényes objektumtípus lehet. Ilyen esetben a *feltétel* akkor **True**, ha az *objektumnév* egy, az *objektumtípus* argumentumban adott objektumtípus. Formailag az utasítássorban az **If** előtt más utasítást nem lehet elhelyezni
- Az *utasításcsoport1* argumentum azt az utasítást tartalmazza, amelyet a *feltétel* **True** értéke esetén kell végrehajtani. Az utasítás lehet **GoTo** szerkezet, továbbá egy vagy több, ugyanabban a sorban egymástól kettősponttal elválasztott utasítás is. Az *utasításcsoport1* végrehajtása után a programfutás a következő sor első utasításával folytatódik.
- A feltételes **Else** *utasításcsoport2* argumentum azokat az utasításokat tartalmazza, amelyeket a *feltétel* **False** értéke esetén kell végrehajtani. Ha az utasítás nem tartalmazza az **Else** *utasításcsoport2* részt, a programfutás a következő sor első utasításával folytatódik.

Valamelyest összetettebb a Szintaxis 2 szerinti utasításstruktúra, amelyben az utasításblokkok közül a feltételektől függően csak az egyik kerül végrehajtásra.

- Az **If** *feltétel1* **Then** argumentum itt is **True** vagy **False** értékű kifejezés.

- A feltételes *utasításcsoport1* argumentum azt vagy azokat az utasításokat tartalmazza, amelyeket a *feltétel1* **True** értéke esetén kell végrehajtani. Az *utasításcsoport1* végrehajtása után a programfutás az **End If** utasítást követő sorban folytatódik.
- Ha a *feltétel1* kiértékelése **False** eredményt adott, a feltételes **ElseIf** *feltétel2* **Then** argumentumban szereplő kifejezés értékétől függően kerül végrehajtásra a következő, feltételes *utasításcsoport2* utasításcsoport vagy a következő, ugyancsak feltételes **ElseIf** *feltétel3* **Then** feltétel kiértékelése. Ilyen módon tetszés szerinti szintű **If...Then...Else** utasítást lehet egymásba ágyazni, amelyek sorát végül az **End If** utasítással kell lezárni. Ha a teljes láncban az **If** és **ElseIf** utáni feltételek egyike sem teljesül, a feltételes **Else** *utasításcsoport_n* utasítás végrehajtása után az **End If** utasítást követő sorba kerülünk.

Az **If...Then...Else** utasításban az **If** előtt semmi, az **Else**, **ElseIf** és **End If** előtt pedig legfeljebb címke vagy sorszám szerepelhet.

A Szintaxis 2 használata során vigyázni kell arra, hogy az **If...Then...Else** a **Then** után megjelenő bármely utasítás esetén a Szintaxis 1 szerint fog működni.

Az **If...Then...Else** legtöbb esetben a **Select Case** utasítással helyettesíthető, azonban ott az **IfTypeOf** *objektumnév* **Is** *objektumtípus* szerkezet nem használható.

Input#

Szintaxis: **Input #** *fájlszám*, *változólista*

Az utasítás egy **Input** vagy **Binary** módban megnyitott soros fájlból egy vagy több adatot olvas be az adott változókba.

- A beolvasandó adatot tartalmazó fájlt a megnyitásakor az **Open** utasításban használt számmal azonos értékű *fájlszám* argumentummal jelölhetjük ki.
- A *változólista* argumentum azon változók vesszővel elválasztott felsorolása, amelyekbe a soros fájlból beolvasott adatokat helyezzük. A *változólista* nem jelölhet ki tömböt vagy objektumváltozót, azonban egy tömb elemeit leíró változó vagy egy felhasználó által definiált típus használata megengedett.

Az olvasáskor a fájlban szereplő standard karakteres és numerikus adatok változtatás nélkül kerülnek a változókba, az adatok közötti " karakterek azonban nem kerülnek beolvasásra. A beolvasott adatok kezelése a következő táblázatnak megfelelően történik:

Ha az adat:	A változó értéke:
Üres sor vagy adatelválasztó vessző	**Empty**
#NULL#	**Null**
#TRUE# vagy #FALSE#	**True** vagy **False**

#yyyy-mm-dd hh:mm:ss#	A kifejezés-szerinti dátum és/vagy időpont
#ERROR *hibaszám*#	*hibaszám* (hibaként jelölt **Variant**)

A fájlból beolvasott és a *változólista* argumentumban felsorolt változók számának, típusának és sorrendjének természetszerűleg egyezniük kell. Ha a változó numerikus és a beolvasott adat nem, a változó 0 értéket kap. Amennyiben az olvasás befejezése előtt érünk a fájl végére, a beolvasás hibajelzéssel ér véget.

Az adatbeolvasás megbízhatósága megkívánja, hogy az adatok fájlba írása a **Print#** helyett a **Write#** utasítással történjen, amely biztosítja az adatmezők megfelelő elválasztását.

Kill

Szintaxis: **Kill** ***pathname***

Az utasítás törli a névvel hivatkozott ***pathname*** argumentumban megadott fájlt vagy fájlokat. A ***pathname*** egy karakteres kifejezés, amely a meghajtónevet és elérési utat is tartalmazhatja. A karakteres kifejezésben a szokásos módon használhatók a helyettesítő, un. wildcard (**?** és *****) karakterek, így egyetlen utasítással fájlcsoportok törlése is lehetséges.

Törlendő fájlként már megnyitott fájlt kijelölve hibajelzést kapunk. Az utasítás csak fájlok törlésére használhatók, a könyvtárak törlése az **RmDir** utasítással lehetséges.

Let

Szintaxis: [**Let**] *változó* = *kifejezés*

Ez az úgynevezett értékadó utasítás a *változó* argumentumban adott nevű változónak vagy tulajdonságnak a *kifejezés* argumentum értékét adja. Az utasításban a **Let** kulcsszó feltételes, elhagyható.

Természetesen a *változó* és a *kifejezés* típusainak egymással kompatibilisnek kell lennie egyébként futásidejű `"Type mismatch"` hibajelzést kapunk. A **Variant** adatok numerikus vagy **String** típusúnak is beállíthatók, azonban ez fordítva nem minden esetben érvényes. A **Null** kivételével bármely **Variant** adható **String** változónak, de numerikus változó csak számként értelmezhető **Variant** lehet. (A kifejezések számként értelmezhetősége az **IsNumeric( )** függvénnyel ellenőrizhető.)

Az utasítást rekordváltozók (összetett, a felhasználó által definiált típusú változók) esetében csak akkor lehet használni, ha mindegyik változó ugyanabba a felhasználó által definiált típusba tartozik. A különböző felhasználó által definiált típusú rekordváltozók beállítására az **LSet** utasítás, az objektumhivatkozások változókba beállítására pedig a **Set** utasítás használható.

Line Input

Szintaxis: **Line Input #** *fájlszám*, *változónév*

Ez az utasítás egyetlen sort olvas be egy megnyitott soros fájlból és azt egy **String** változóba helyezi el.

- A *fájlszám* argumentum a beolvasandó sort tartalmazó fájlt az **Open** megnyitására használt fájlszám alapján jelöli ki. A sor beolvasása a kocsivissza, azaz CHR$(13) vagy a kocsivisssza-soremelés CHR$(13) + CHRS(10) karakterekig tart.
- A beolvasott sor a *változónév* argumentumban adott érvényes nevű karakteres változóba kerül.

Lock...Unlock

Szintaxis:

Lock [#] *fájlszám* [, *rekordtartomány*]
...
Unlock [#] *fájlszám* [, *rekordtartomány*]

A **Lock...Unlock** utasítással a több feldolgozási eljárás által is elérhető fájlokat használó környezetben egy megnyitott fájl egészének vagy egy részének elérése írható elő.

- A kötelező *fájlszám* argumentumban a fájt az **Open** utasítással történő megnyitásakor használt számmal kell kijelölni.
- Az utasításban a feltételes *rekordtartomány* a *rekordszám* vagy az [*első*] **To** *utolsó* alakban használható, ahol az argumentumok értelmezése a következő:
  - A *rekordszám* argumentum a fájlon belül a hozzáférés elöli zárolás vagy a felszabadítás kezdetét jelöli ki: a **Random** módban megnyitott fájloknál a rekordszámmal, a **Binary** módban megnyitottaknál pedig byte-számmal. Ha hiányzik a *rekordszám* argumentum, az utasítás a teljes fájlra érvényes.
  - Az [*első*] **To** *utolsó* alakban egy rekord vagy byte-tartomány jelölhető ki, amelynek kezdetét az *első*, végét pedig az *utolsó* argumentum határozza meg. Ha hiányzik az *első* argumentum, az utasítás az elsőtől az utolsó rekordig vagy byte-ig számig érvényes.

Az utasításpárban a **Lock** és az **Unlock** argumentumainak tökéletesen egyezőknek kell lenniük.

A **Lock...Unlock** utasítás — függetlenül az *első* és az *utolsó* értékétől — a teljes fájlra terjed ki, ha az szekvenciális olvasásra vagy írásra van megnyitva.

A fájlok bezárása vagy a program befejezése előtt minden zárolást fel kell oldani.

LSet

Szintaxis 1: **LSet** *karakteres_változó* = *karakteres_kifejezés*

Szintaxis 2: **LSet** *rekordváltozó1* = *rekordváltozó2*

Az **LSet** utasítás egy karakteres kifejezést egy változón belül balra igazítva helyez el (Szintaxis 1), vagy egy felhasználó által definiált típusú változót egy másik, ugyancsak felhasználó által definiált típusú változóba másol (Szintaxis 2).

Az utasítás első (Szintaxis 1) alakjának argumentumai a következők:

- A *karakteres_változó* argumentumban annak a változónak a neve szerepel, amelybe balra igazítottan a kell a karakteres kifejezést elhelyezni.
- A *karakteres_kifejezés* argumentumban annak a karakteres kifejezésnek kell lennie, amelyet a változóba akarunk elhelyezni. Ha a *karakteres_kifejezés* hosszabb, mint a *karakteres_változó*, a kifejezés bal oldaláról csak a változó hosszúságának megfelelő számú karakter kerül a változóba.

Az utasítás második (Szintaxis 2) alakjában az argumentumok értelmezése a következő:

- A *rekordváltozó1* argumentumban annak a felhasználó által definiált típusú változónak kell lennie, amelybe a másolás történik.
- A *rekordváltozó2* argumentumban a másolandó, ugyancsak egy felhasználó által definiált típusú változónak kell szerepelnie. A felhasználó által definiált, de eltérő típusú változók közötti másolást az előre nem látható eredmény miatt kerülni kell: ilyenkor az argumentumok adattípusaira tekintet nélkül a *rekordváltozó2* bináris adatai egyszerűen csak a *rekordváltozó1* memóriaterületére másolódnak.

Mid

Szintaxis: **Mid(** *karakteres_változó*, *első* [, *hossz*]) = *karakteres_kifejezés*

A **Mid** utasítással egy karakteres kifejezést vagy annak egy részét egy másik karakteres kifejezéssel vagy annak egy részével lehet helyettesíteni.

A *karakteres_változó* argumentumba az a karakterlánc kerül, amelyet helyettesíteni kívánunk.

Az *első* argumentum azt írja elő, hogy a helyettesítés a *karakteres_változó* hányadik karakterpozíciójától kezdődjön.

A *karakteres_változó* argumentumba az adott pozíciótól kezdődően a *karakteres_kifejezés* argumentum tartalma kerül.

Ha a *karakteres_változó* csak egy részét akarjuk helyettesíteni, illetve a *karakteres_kifejezés* csak meghatározott számú karakterét kívánjuk az eredeti karaktervváltozóba helyezni, helyettesítendő karakterek számát a feltételes *hossz* argumentum szabja meg.

A helyettesítés során a *karakteres_változó* eredeti hosszúsága nem nőhet, függetle-

nül a *karakteres_kifejezés* hosszától vagy az *első* és a *hossz* értékétől.

MidB

Szintaxis: **MidB(** *karakteres_változó*, *első* [, *hossz*]) = *karakteres_kifejezés*

A **MidB** a **Mid** utasításhoz hasonló funkciójú, de **Byte** típusú adatokat tartalmazó karakteres kifejezések esetén használatos, és az utasításban az *első* és a *hossz* argumentumok nem karakterekre vonatkoznak, hanem byte-pozícióként vagy byte-számként értelmezettek.

MkDir

Szintaxis: **MkDir** *elérési_út*

A **MkDir** utasítás egy új könyvtárat hoz létre. Az *elérési_út* argumentum egy

[*meghajtó*:\ ] *könyvtár* [*könyvtár1*]...

alakú karakteres kifejezés lehet. A feltételes *meghajtó* argumentumot elhagyva a könyvtár az aktív meghajtón jön létre.

A szóközt tartalmazó nevű könyvtárak tartalmához bizonyos alkalmazások hozzá tudnak férni, azonban törlésük az operációs rendszerek szokásos parancsaival nem lehetséges, erre a célra az **RmDir** utasítást kell használni.

Name

Szintaxis: **Name** *régi_ elérési_út* **As** *új_ elérési_út*

Ezzel az utasítással egy meglévő fájl neve módosítható, illetve egy létező fájt ugyanazon meghajtón az egyik könyvtárból egy másikba lehet átvinni.

Az utasításban a *régi_ elérési_út* és az *új_ elérési_út* argumentumok egyaránt karakteres kifejezések lehetnek, ahol a *régi_ elérési_út* a létező, átnevezendő fájt jelöli ki, míg az *új_ elérési_út* az új fájlnevet és annak helyét írja elő.

Az elérési utak feltételesen tartalmazhatják a meghajtó nevét is, azonban az átnevezett fájlnak a régivel azonos meghajtón kell maradnia. Ha az elérési utakból a meghajtónév és könyvtárnév hiányzik, az utasítás az éppen érvényes könyvtáron belül a fáj átnevezésére, míg azonos fájlnevet, de eltérő könyvtárnevet megadva az adott fájl könyvtárak közötti mozgatására használható.

Az utasítás nyitott fájlra történő alkalmazása hibát okoz, a nyitott fájt átnevezés vagy mozgatás előtt be kell zárni.

On Error

Szintaxis: **On Error {GoTo** *sor* **| Resume Next | GoTo 0}**

Az utasítással a programfutás alatt egy hibakezelő rutint engedélyezhető és aktív/inaktív állapotba lehet kapcsolni, valamint ki lehet jelölni a hibakezelő rutinnak az eljáráson belüli helyét. Az utasítás használata nélkül minden futásidejű hiba fatális lenne, azaz hibajelzés után a programfutás megállna.

Az **On Error** utasítás három alakban használható:

Ha az **On Error** alakja:	Hatása:
On Error GoTo *sor*	A programfutás alatti hiba esetén engedélyezi a *sor* argumentumban adott címkével vagy sorszámmal jelölt sorban kezdődő hibakezelő rutint. A sornak az **On Error** utasítással azonos eljárásban kell lennie, különben fordításidejű hiba következik be. Az objektumok használatakor inkább az **On Error Resume Next** alak javasolt.
On Error Resume Next	A programfutás alatti hiba esetén a vezérlést a hibát okozó utasítás utáni sorra, vagy ha a hiba egy meghívott eljárásban következett be, akkor az azt meghívó utasítás utáni sorra adja.
On Error GoTo 0	Az éppen futó eljárásban letilt minden hibakezelő eljárást.

Engedélyezett hibakezelőnek az **On Error** utasítással bekapcsolt rutint, aktívnak pedig az éppen egy hibát lekezelő rutint nevezzük. Ha azalatt következik be egy hiba, amíg egy hibakezelő rutin aktív, az éppen futó eljárás hibakezelő rutinja nem tudja a hibát lekezelni — a vezérlésnek vissza kell adódnia a hívó eljárásnak és ha annak van engedélyezett hibakezelője, az aktívvá válva fogja lekezelni a hibát. Amikor a hibakezelő a vezérlést egy hívó eljárásnak visszaadja, az adott eljárás válik az aktuális eljárássá. Abban az esetben, ha a hívó eljárás hibakezelője is aktív, a vezérlés mindaddig visszalép egy hívóeljárással, amíg egy engedélyezett, de nem aktív hibakezelőt talál. A hiba sikeres lekezelése után a vezérlés ismét visszaadódik az aktuális eljárásnak a **Resume** által kijelölt pontjára. Ha nem található engedélyezett, de nem aktív hibakezelő, a hiba a bekövetkezésének helyén fatálisnak minősül.

A hibakezelő rutinok a hiba okát az **Err** objektum **Number** tulajdonságának értéke alapján határozzák meg és még azelőtt vizsgálhatják vagy menthetik ezeket az értékeket, mielőtt bármely más hiba következhetne be vagy mielőtt egy esetleg hibát okozó eljárás hívása történhetne. Az **Err** objektum tulajdonságai csak az utolsóként bekövetkezett hiba értékeit tükrözik. Az **Err.Number** tulajdonsághoz tartozó hibaüzenetet az **Err.Description** tartalmazza.

Az **On Error Resume Next** utasítás a program végrehajtását vagy a futásidejű hibát okozó utasítást, vagy az **On Error Resume Next** utasítást tartalmazó eljáráson kívüli legutolsó hívást követő utasítással folytatja, vagyis a bekövetkezett hiba ellenére folytat-

ható a program futtatása. Ilyenkor az eljáráson belül folyamatosan helyezhető el a hibakezelő rutin és szükségtelenné válik a vezérlés átadása az eljáráson belüli másik helyre. Az **On Error Resume Next** utasítás egy másik eljárás hívásakor inaktívvá válik, vagyis végrehajtható minden egyes hívott rutinban, ha a folyamatos hibakezelést kívánunk megoldani.

Az **On Error Resume Next** utasítás használata az **On Error GoTo** utasításnál kedvezőbb, mivel az objektumokhoz történő hozzáférés során bekövetkező hibák esetén lehetővé teszi a hibaszámot visszaadó objektum egyértelmű azonosítását. Minden egyes objektumkapcsolat után az **Err** objektum vizsgálatával azonnal meghatározható, hogy melyik objektumhoz történő hozzáférés során következett be hiba, melyik objektum állította be a hibaszámot az **Err.Number** tulajdonságba és eredetileg melyik okozta az **Err.Source** által meghatározott hibát.

Az **On Error GoTo 0** alakú, az éppen futó eljáráson belüli hibakezelést letiltó utasításban a **GoTo 0** csak szimbolikus jelentésű, ténylegesen még akkor sem egy sor számát jelöli, ha az adott eljárásban létezik 0 sorszámú sor. Az **On Error GoTo 0** utasítás nélkül a hibakezelő az eljárás befejeződésekor automatikusan kikapcsolttá válik.

A hibakezelő rutinnak csak egy adott hiba bekövetkeztekor szabad futnia, amit például **Exit Sub** és **EndSub**, **Exit Function** és **End Function** vagy **Exit Property** és **End Property** utasítások közé helyezésükkel lehet biztosítani.

Amikor egy OLE Automatizmus objektum futtatható fájlként fut, a benne bekövetkező, belsőleg nem azonosított hiba a vezérlő alkalmazásba, míg a fejlesztési környezetben (ha az Options párbeszédpanel Advanced fülén található Break In OLE Server lehetőség nem kiválasztott állapotú) az OLE Automatizmus vezérlőelemnek adódik vissza.

Az olyan OLE Automatizmus objektum írásakor, amely más objektumokhoz is hozzáférhet, a hibákat kezeletlenül is megkísérelhetjük visszaadni. Ha az ilyen hibákat nem tudjuk lekezelni, az objektumunk azt a hibaszámnak az **Err.Number** tulajdonságba saját hibaként történő elhelyezése után visszaadhatja az objektum meghívójának. Saját hibát a **vbObjectError** szimbolikus konstanshoz a saját hibaszám hozzáadásával definiálhatunk.

A 32-bites Microsoft Windows operációs rendszerben a DLL hívások alatti rendszerhibák nem növelik a kivételeket és a Visual Basic belső hibafigyelésével nem figyelhetők. A DLL függvények hívásakor meg kell vizsgálni minden egyes visszaadott érték API előírások szerinti sikerességét vagy sikertelenséget, és egy hiba esetében az **Err** objektum **LastDLLError** tulajdonság értékét.

On...GoSub

Szintaxis: **On** *kifejezés* **GoSub** *céllista*

Ez az úgynevezett feltételes vezérlésátadó utasítás egy kifejezés értékétől függően egy vagy több sor utasításait hajtja végre.

- A programfutást a *kifejezés* argumentumban megadott, numerikus kifejezés értéke vezérli. A *kifejezés* kiértékelés után egész számra kerekítődik.
- A vezérlés a *kifejezés* értékétől függően a *céllista* argumentumban adott számú vagy címkéjű sorban található szubrutinra adódik át. A *céllista* egy vagy több, egymástól vesszővel elválasztott elemet tartalmazhat. A listában sorszámok és címkék keverten is használhatók, számuk nincs korlátozva. Ha a *céllista* hossza egy sornál hosszabb, a sor végén sorfolytonosság karaktert (_) kell használni.

Ha a *kifejezés* értéke az érvényes 1 — 255 közötti tartományon kívülre esik, hatása a következő:

Ha a *kifejezés* értéke:	Hatása:
0	A vezérlés a következő utasításra lép.
programlistában szereplőknél nagyobb	A vezérlés a következő utasításra lép.
<0	Hibajelzés (`Illegal function call`).
>255	Hibajelzés (`Illegal function call`).

On...GoTo

Szintaxis: **On** *kifejezés* **GoTo** *céllista*

Az utasítás csak annyiban tér el az **On...GoSub** utasítástól, hogy a programfutás a *kifejezés* értékétől függően a *céllista* argumentumban adott számú vagy címkéjű sorban folytatódik. A használatra vonatkozó egyéb tudnivalók azonosak az **On...GoSub** utasításnál leírtakkal.

Open

Szintaxis: **Open** *elérési_út* [**For** *megnyitási_mód*] [**Acces** *elérési_mód*] [*engedélyezett_elérés*] **As** [#] *fájlszám* [**Len** = *rekordhossz*]

Az utasítás egy fájlt a ki- vagy beviteli műveletek számára elérhetővé tesz, puffer tárolóterületet képez a fájlműveletek számára, és kijelöli a nyitott fájl elérési módját.

- A megnyitandó fájl nevét a meghajtó- és könyvtárnévvel is kibővíthető alakban az *elérési_út* argumentum karakteres kifejezésben kell megadni. Ha az adott fájl még nem létezik, azt az **Open** utasítás az **Append**, **Binary**, **Output** vagy **Random** módú megnyitás esetén automatikusan létrehozza.
- Az **As** [#] *fájlszám* argumentum a fájl megnyitáshoz használt 1 és 511 közötti azonosítószáma, amellyel az adott fájlra annak lezárásig más utasításokban lehet hivatkozni. Az éppen következő szabad fájlszámot a **FreeFile( )** függvénnyel lehet lekérdezni.

Az utasításban használható feltételes argumentumok szerepe a következő:

- A **For** *megnyitási_mód* argumentumban megszabható hogy az adott fájt milyen módra kívánjuk megnyitni. A választható módok: **Append**, **Binary**, **Input**, **Output** vagy **Random**. Más processz által már megnyitott fáj más fájlszámmal ismét megnyitható **Binary**, **Input** vagy **Random** módban, de **Append** és **Output** módban csak előzőleg lezárt fájlt lehet megnyitni, különben hiba következik be.
- Az **Access** *elérési_mód* argumentum a megnyitott fájlon engedélyezett műveleteket szabja meg: a **Read** olvasási, a **Write** írási, a **Read Write** pedig írási és olvasási műveletet engedélyez.
- Az *engedélyezett_elérés* argumentum a megnyitott fájlnak más processzek számára történő hozzáférését szabályozza: a **Shared** megosztott elérést engedélyez, a **Lock Read**, **Lock Write** és a **Lock Read Write** pedig a fájl olvasásának, írásának vagy írásának és olvasásának más processzek előli lezárását jelenti.
- A **Len** = *rekordhossz* argumentum 32767-nél nem nagyobb értékű lehet és a **Random** módban megnyitott fájlok esetén a fájl rekordhosszát, sorrendi hozzáférésre megnyitott fájlok esetén pedig a belső karakterpuffer méretét szabja meg. A **Binary** megnyitási módban hatástalan záradék.

Option Base

Szintaxis: **Option Base {0 | 1}**

Ez a modulszinten használt utasítás a tömbváltozók indexének kezdeti értékét módosítja, illetve állítja be.

Az utasítás **Option Base 0** alakjában a tömbindex nulláról indul, azaz a tömb első tagja a nulla sorszámot kapja, az **Option Base 1** alakban pedig a tömb első tagja az egyes sorszámú lesz. Az **Option Base** utasítás explicit használata nélkül az **Option Base 0** az alapértelmezett tömbindexelés mód.

Az **Option Base** utasítás helye a programban a változókat definiáló vagy konstansokat deklaráló utasítások előtt van, egy modulon belül csak egyszer lehet használni és csak az adott modulon belüli tömbökre van hatása.

Az **Option Base 1** utasítás helyett egy tömb első elemének beállítására **To** záradékkal használt **Dim**, **Private**, **Public**, **ReDim** és **Static** utasítások rugalmasabb módot kínálnak, azonban ilyen esetben a **To** záradékkal explicit módon nem beállított alsó index az alapértelmezett 0 érték lesz és azt az **Option Base 1** utasítással nem lehet módosítani.

Option Compare

Szintaxis: **Option Compare {Binary | Text}**

Ez a modulszintű, a deklarációs részben a változókat definiáló vagy konstansokat deklaráló utasítások előtt elhelyezhető utasítás az adott modulban a karakteres adatok érvényes összehasonlítási módját állítja be.

Az **Option Compare Binary** alak esetén az összehasonlítás a karakterek bináris, a kódlapoknak megfelelő sorrendjén alapul, míg az **Option Compare Text** alak esetén a területi beállításnak megfelelő, a karakterek kis- és nagybetűs alakját azonosnak tekintő, karakterszerinti összehasonlítás történik.

Az **Option Compare** utasítás explicit használata nélkül az **Option Compare Binary** az alapértelmezett összehasonlítási mód.

Option Explicit

Szintaxis: **Option Explicit**

Ez a modulszintű, a deklarációs részben a változókat definiáló vagy konstansokat deklaráló utasítások előtt elhelyezhető utasítás az adott modulban használt minden változó explicit, a **Dim**, **Private**, **Public**, **ReDim** vagy **Static** utasítások valamelyikével történő deklarálását írja elő. Deklarálatlan változó használatát megkísérelve fordításidejű hibajelzés (`Variable not declared`) következik be.

Az **Option Explicit** utasítás használata nélkül minden változó alapértelmezetten **Variant** adattípusú lesz, kivéve a **Def***típus* utasítással eltérően deklarált változókat.

Az **Option Explicit** utasítás használata megakadályozza a változónevek pontatlan, hibás beírása okozta problémákat.

Option Private

Szintaxis:

Az utasítás csak a Visual Basic for Application objektumtára számára látható, a Visual Basic 4.0-s verzióban nem használható.

Print#

Szintaxis: **Print#** *fájlszám*, [*lista*]

Az utasítás formázott adatokat ír egy soros fájlba.

- A *fájlszám* argumentumban a soros fájlt a megnyitásához az **Open** utasításban használttal azonos számmal lehet kijelölni.
- Az [**Spc**(*n*) | **Tab**(*n*)] [*kifejezés*] [*karakterpozíció*] alakú *lista* argumentum a fájlba írandó kifejezés vagy azok listája lehet. Ha a feltételes *lista* hiányzik, üres sort kapunk. Több kifejezést tartalmazó listában az egyes kifejezéseket egymástól szóközzel vagy pontosvesszővel is el lehet választani. A *lista* feltételes argumentumai a következők:
  - A *kifejezés* argumentumban a soros fájlba írandó adat, azaz egy numerikus vagy karakteres kifejezés szerepel.

- A *kifejezés* tartalmának beillesztési pozícióját két feltételes függvénnyel lehet előírni: az **Spc**(*n*) hatására az adat *n* darab szóköz után, a **Tab**(*n*) hatására pedig az *n*-ik számú oszlopba kerül kiírásra. A **Tab**( ) függvényt üres argumentummal használva a beillesztési pont a következő zóna eleje lesz. Az **Spc**( ) és **Tab**( ) függvények a kifejezéslista egyes tételei mellett is használhatók A *kifejezés* elhagyásakor az utasítás egy üres sort ír a fájlba, de a fájlszámot követő vesszőt ekkor is ki kell tenni!
- Ha a következő kiíratási pozíciót az alapértelmezett következő sor kezdete helyett máshová kívánjuk beállítani, erre a *karakterpozíció* argumentum használható:
  - Ha az argumentum a ";" karakter, a kiírás közvetlenül az előző karakter után folytatódik.
  - A **Tab**(*n*) függvény használatával a kiírást az *n*-ik számú oszlopba lehet folytatni. A **Tab**( ) függvényt üres argumentummal használva a beillesztési pont a következő zóna eleje lesz.

A **Print#** utasítással fájlba írt minden adat megjelenése a területi beállítástól függ:

- A **Boolean** típusú adatok esetén `False` vagy `True` szó kerül a fájlba. A **True** és **False** kulcsszavak a rendszer területi beállítástól függően fordítódnak.
- A **Date** típusú adatok a rendszerben érvényes rövid alakú dátumformában kerülnek a fájlba. Ha az adat dátumot vagy időpontot jelölő része hiányzik vagy nulla, csak a meglévő rész íródik a fájlba.
- Ha a *lista* argumentum **Empty** értékű, a fájlba semmi sem íródik, míg **Null** érték esetén a területi beállításnak megfelelően lefordított `Null` szó kerül bele.
- Az **Error** típusú adatoknál a kimenet `Error` *`hibaszám`* alakú lesz, ahol az **Error** kulcsszó megfelelően lefordítva kerül a fájlba.

Mivel a **Print#** az adatok képét írja a fájlba, az adatokat megfelelően el kell elválasztani egymástól. Ha a következő zóna elejére történő pozícionáláshoz az argumentum nélküli **Tab**( ) függvényt használjuk, a **Print#** utasítás szóközt helyez el a fájl kiírási mezői közé.

Ha az adatokat a későbbiekben az **Input#** utasítással kívánjuk kiolvasni, a fájlba írásukhoz a **Print#** utasítás helyett a **Write#** utasítást használjuk, mivel ez biztosítja az adatok megfelelő elválasztását és bármely területi beállításban helyes kiolvasását is.

Private

Szintaxis: **Private** *változónév* [([*indextartomány*])] [**As** [**New**] *típus*] [, *változónév* [([*indextartomány*])] [**As** [**New**] *típus*]...

Ezzel a modulszinten használatos utasítással **Private** változók vagy objektumok tí-

pusát deklarálhatjuk, egyben a hozzájuk tartozó tárolóterület kiosztását, allokálását is elvégezve. Használatát követően a deklarált változók az adott modulon belüli minden eljárás számára elérhetők lesznek, de más modulokból már nem, azaz a változók az adott modulra lokálisak maradnak.

A **Private** argumentumai a változóneveket és azok típusait, tömbváltozók esetén az indextartományokat is tartalmazó lista.

- A *változónév* argumentum a deklarálandó változónak az általános névadási konvenciók szerint képzett neve.
- A feltételes *indextartomány* argumentum a legfeljebb 60 dimenziós tömbváltozók dimenzióinak egymástól vesszővel elválasztott, [*alsóindex* **To**] *felsőindex* alakú felsorolása. Hiányzó *alsóindex* esetén annak értéke az **Option Base** utasítástól függően 0 vagy 1.
- A használható változók típusait az **As** [**New**] *típus* argumentumban írhatjuk elő. A használható típusok: **Byte**, **Boolean**, **Integer**, **Long**, **Single**, **Double**, **Currency**, **String** (változó hosszúságú karakterlánc), **String** * *hosszúság* (rögzített hosszúságú karakterlánc), **Variant**, **Boolean**, **Date**, **Object**, felhasználói típus vagy konkrét objektumtípus jelölhető meg. Minden egyes változóhoz külön-külön típusmeghatározás tartozik. Az adat- vagy objektívtípus nélkül deklarált változók — hacsak egy **Def***típus* utasítással nincs más típus előírva — alapértelmezetten **Variant** típusúak lesznek.
    - A típusdeklarációban a feltételes **New** kulcsszóval azt jelezhetjük, hogy a deklarált objektumváltozó egy Visual Basic objektum egy új példánya vagy egy külsőleg létrehozható OLE Automatizmus objektum. Ha az objektumváltozó deklarációjában nem használt a **New**, az objektumnak nem létezik példánya. A **New** kulcsszó nem használható belső adattípusú változó vagy függő OLE Automatizmus objektumok példányainak deklarálására. Az objektumváltozó csak létező, használat előtt a **Set** utasítással értéket kapott objektumra hivatkozhat, egyébként a deklarált objektumváltozó értéke **Nothing**.

Dinamikus tömb deklarálása a **Private** utasításban üres *indextartomány* megadásával történik. A deklarálás után a dinamikus tömbök dimenziójának és elemeinek száma eljárásszinten a **ReDim** utasítással definiálhatók. A **Dim**, **Private** és **Public** utasítással explicit módon létrehozott tömbváltozó újradimenzionálása hibát okoz.

A változók inicializálásakor a **Private** utasítás minden numerikus változót és tömböt nulláz, a rögzített hosszúságú karakterláncot nullákkal, a változó hosszúságút üres ("") karakterláncokkal, a **Variant** típusúakat pedig **Empty** értékekkel tölti fel. A felhasználó által definiált változótömbök az egyedi változókkal azonos módon kapnak kiinduló értéket.

Mint minden eljárásszintű deklarációt, a **Private** utasítást is az eljárás elején szokás elhelyezni.

Property Get

Szintaxis:

[Private | Public] [Static] Property Get *eljárásnév* [(*argumentumlista*)] [**As** *típus*]
[*utasításcsoport1*]
[*eljárásnév* = *kifejezés*]
[**Exit Property**]
[*utasításcsoport2*]
[*eljárásnév* = *kifejezés*]
End Property

Ez az utasítás egy tulajdonság értékét visszaadó **Property** eljárást alkotó változókat, argumentumokat és programkódot deklarál.

- A feltételes kulcsszavak közül **Private** azt írja elő, hogy a **Property Get** eljárása kizárólag csak az azt tartalmazó modulon belüli eljárások számára legyen elérhető. Külön deklaráció nélkül ez az alapértelmezett.
- Ha globális, minden modul minden eljárása által használható **Property** eljárást akarunk létrehozni, a **Public** kulcsszót kell használni.
- A **Static** azt jelzi, hogy a **Property Get** helyi változói a hívások között is megtartják értéküket. Ha nem használjuk ezt a deklarációt, a változóértékek a hívások között elvesznek. A **Static** nem hat az eljárásban használt, azonban a **Property Get** utasításon kívül deklarált változókra.
- A **Property Get** eljárásának nevét az *eljárásnév* argumentumban lehet megadni. A név képzésére a szokásos konvenciók vonatkoznak azzal az eltéréssel, hogy a modulban azonos nevű **Property Let** vagy **Property Set** is megengedett.
- Az eljárás nevét a feltételes *argumentumlista*, azaz az eljárásban használt, annak meghívásakor átadott változó (vagy vesszővel elválasztott változók) zárójelbe tett listája követi a [**ByVal** | **ByRef**] *változónév* [( )] [**As** *típus*] alakban.
  - Az argumentumok átadási módját a **ByVal** vagy **ByRef** kulcsszavakkal lehet megszabni, a **ByVal** az értékkel, az alapértelmezetten érvényes **ByRef** pedig a címük alapján, hivatkozással történő értékátadást eredményez.
  - A *változónév* argumentumban a változó nevét kell megadni. Az egyes argumentumok nevének és típusának egyezőnek kell lennie a **Property Let** utasításban (ha van ilyen) használtakkal. Ha egy változó adattömb, ezt a tömb indextartományának megadása nélkül, a *változónév*( ) alakkal kell jelezni.
  - Az átadott változók típusai a feltételes **As** *típus* argumentumban deklarálhatók, ahol a *típus* **Byte**, **Boolean**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, **Single**, **String** (csak változó hosszúságú), **Variant**, felhasználó által definiált típus vagy objektumtípus lehet.

- A **Property Get** eredményének típusát általában a használt argumentumok típusa dönti el, azonban a feltételes **As** *típus* argumentumban explicit módon is deklarálhatók. A *típus* **Byte**, **Boolean**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, **Single**, **String** (csak változó hosszúságú), **Variant** vagy felhasználó által definiált típus lehet. Semmilyen tömböt sem kaphatunk vissza, azonban tömböt tartalmazó **Variant** adatot igen. A **Property Get** eredményének típusaként a **Property Let** utasításban (ha van ilyen) használttal azonos típust kell megadni.
- A **Property Get** első lépése az *utasításcsoport1* utasítás vagy utasításcsoport (általában valamilyen vizsgálat) végrehajtása.
- A **Property Get** az *utasításcsoport1* kiértékelése és a visszaadandó érték beállítása után vagy az *utasításcsoport2* végrehajtását kezdi meg vagy a tetszőleges számban elhelyezhető **Exit Property** utasításon keresztül azonnal a **Property Get** eljárását hívó utasítás utáni utasításra lép.
- A **Property Get** által visszaadott értékét a feltételes *eljárásnév* = *kifejezés* határozza meg. Ha nem történik eredmény beállítása, az *eljárásnév* eljárás meghívásának eredménye numerikus típusú eljárásnál 0, karakteres típusúnál üres karakterlánc, **Variant** típus esetében pedig **Empty** érték lesz.
- A **Property Get** eljárását az **End Property** zárja le.

Mivel a **Property Get** egy értéket ad vissza — eltérően a hasonló **Sub** és **Property Let** utasításoktól — ugyanúgy használhatjuk egy kifejezés jobb oldalán, mint egy **Function** eljárást vagy egy tulajdonság értékét. A **Property Get** nem használható más **Property**, **Sub** vagy **Function** eljárásokon belül.

Property Let

Szintaxis:

[Private | Public] [Static] Property Let *eljárásnév* [(*argumentumlista*)]
[*utasításcsoport1*]
[**Exit Property**]
[*utasításcsoport2*]
End Property

Ez az utasítás egy tulajdonság értékét beállító **Property** eljárást alkotó változókat, argumentumokat és programkódot deklarál.

- A feltételes kulcsszavak közül **Private** azt írja elő, hogy a **Property Let** eljárása kizárólag csak az azt tartalmazó modulon belüli eljárások számára legyen elérhető. Külön deklaráció nélkül ez az alapértelmezett.
- Ha globális, minden modul minden eljárása által használható eljárást akarunk létrehozni, a **Public** kulcsszót kell használni.
- A **Static** azt jelzi, hogy a **Property Let** helyi változói a hívások között is meg-

tartják értéküket. Ha nem használjuk ezt a deklarációt, a változóértékek a hívások között elvesznek. A **Static** nem hat az eljárásban használt, azonban a **Property Let** utasításon kívül deklarált változókra.

- A **Property Let** eljárásának nevét az *eljárásnév* argumentumban lehet megadni. A név képzésére a szokásos konvenciók vonatkoznak, itt azzal az eltéréssel, hogy ugyanebben a modulban azonos nevű **Property Get** vagy **Property Set** is megengedett.
- Az eljárás nevét a feltételes *argumentumlista*, azaz az eljárásban használt, annak meghívásakor átadott változó (vagy vesszővel elválasztott változók) zárójelbe tett listája követi a [**ByVal** | **ByRef**] *változónév* [( )] [**As** *típus*] alakban. A **Property Let** utasításnak legalább egy argumentumot kell tartalmaznia.
  - Az argumentumok átadási módját a **ByVal** vagy **ByRef** kulcsszavakkal lehet megszabni, a **ByVal** az értékkel, az alapértelmezetten érvényes **ByRef** pedig a címük alapján, hivatkozással történő értékátadást eredményez.
  - A *változónév* argumentumban a változó nevét kell megadni. Az argumentumlistában utolsó változó kivételével az egyes argumentumok nevének és típusának egyezőnek kell lennie a **Property Get** utasításban (ha van ilyen) használtakkal. Az argumentumlistában az utolsó argumentum az a tulajdonságérték, amelyet a tulajdonságnak egy kifejezés jobb oldalán adunk át. Ha egy változó adattömb, ezt a tömb indextartományának megadása nélkül, *változónév*( ) alakkal kell jelezni.
  - Az átadott változók típusai a feltételes **As** *típus* argumentumban deklarálhatók, ahol a *típus* **Byte**, **Boolean**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, **Single**, **String** (csak változó hosszúságú), **Variant**, felhasználó által definiált típus vagy objektumtípus lehet. Az argumentumlistában utolsó változónak a **Property Get** utasítás által visszaadandó típussal kell azonosnak lennie.
- A **Property Let** első lépése az *utasításcsoport1* utasítás vagy utasításcsoport (általában valamilyen vizsgálat) végrehajtása.
- A **Property Let** az *utasításcsoport1* kiértékelése után vagy az *utasításcsoport2* végrehajtását kezdi meg vagy a tetszőleges számban elhelyezhető **Exit Property** utasításon keresztül azonnal a **Property Let** eljárását hívó utasítás utáni utasításra lép.
- A **Property Let** eljárását az **End Property** zárja le.

A **Property Let** utasítás — eltérően a hasonló **Function** és **Property Get** utasításoktól — nem ad vissza értéket, ezért csak egy kifejezés bal oldalán szerepelhet egy tulajdonság értékét meghatározó kifejezésben vagy **Let** utasításban. A **Property Let** nem használható más **Property**, **Sub** vagy **Function** eljárásokon belül.

Property Set

Szintaxis:

[Private | Public] [Static] Property Set *eljárásnév* [(*argumentumlista*)]
[*utasításcsoport1*]
[Exit Property]
[*utasításcsoport2*]
End Property

Ez az utasítás egy objektumhivatkozást beállító **Property** eljárást alkotó változókat, argumentumokat és programkódot deklarál.

- A feltételes kulcsszavak közül **Private** azt írja elő, hogy a **Property Set** eljárása kizárólag csak az azt tartalmazó modulon belüli eljárások számára legyen elérhető. Külön deklaráció nélkül ez az alapértelmezett.
- Ha globális, minden modul minden eljárása által használható eljárást akarunk létrehozni, a **Public** kulcsszót kell használni.
- A **Static** azt jelzi, hogy a **Property Set** helyi változói a hívások között is megtartják értéküket. Ha nem használjuk ezt a deklarációt, a változóértékek a hívások között elvesznek. A **Static** nem hat az eljárásban használt, azonban a **Property Set** utasításon kívül deklarált változókra.
- A **Property Set** eljárásának nevét az *eljárásnév* argumentumban lehet megadni. A név képzésére a szokásos konvenciók vonatkoznak azzal az eltéréssel, hogy a modulban azonos nevű **Property Get** vagy **Property Let** is megengedett.
- Az eljárás nevét a feltételes *argumentumlista*, azaz az eljárásban használt, annak meghívásakor átadott változó (vagy vesszővel elválasztott változók) zárójelbe tett listája követi a [**ByVal** | **ByRef**] *változónév* [( )] [**As** *típus*] alakban. A **Property Set** utasításnak legalább egy argumentumot kell tartalmaznia.
  - Az argumentumok átadási módját a **ByVal** vagy **ByRef** kulcsszavakkal lehet megszabni, a **ByVal** az értékkel, az alapértelmezetten érvényes **ByRef** pedig a címük alapján, hivatkozással történő értékátadást eredményez.
  - A *változónév* argumentumban a változó nevét kell megadni. Az argumentumlistában az utolsó argumentum az az objektumhivatkozás, amelyet egy beállítás jobb oldalán adunk át. Ha egy változó adattömb, ezt a tömb indextartományának megadása nélkül, *változónév()* alakkal kell jelezni.
  - Az átadott változók típusai a feltételes **As** *típus* argumentumban deklarálhatók, ahol a *típus* **Byte**, **Boolean**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, **Single**, **String** (csak változó hosszúságú), **Variant**, felhasználó által definiált típus vagy objektumtípus lehet. Az argumentumlistában utolsó változónak a **Property Get** utasítás által visszaadandó típussal kell azonosnak lennie.

- A **Property Set** első lépése az *utasításcsoport1* utasítás vagy utasításcsoport (általában valamilyen vizsgálat) végrehajtása.
- A **Property Set** az *utasításcsoport1* kiértékelése után vagy az *utasításcsoport2* végrehajtását kezdi meg vagy a tetszőleges számban elhelyezhető **Exit Property** utasításon keresztül azonnal a **Property Set** eljárását hívó utasítás utáni utasításra lép.
- A **Property Set** eljárását az **End Property** zárja le.

A **Property Set** utasítás — eltérően a hasonló **Function** és **Property Get** utasításoktól — nem ad vissza értéket, ezért csak egy kifejezés bal oldalán szerepelhet egy objektumtulajdonságot beállító **Set** utasításban. A **Property Set** nem használható más **Property**, **Sub** vagy **Function** eljárásokon belül.

Public

Szintaxis: **Public** *változónév* [([*indextartomány*])] [**As** [**New**] *típus*] [, *változónév* [([*indextartomány*])] [**As** [**New**] *típus*]] ...

Ezzel a modulszinten használatos utasítással **Public** változók vagy objektumok típusát deklarálhatjuk, egyben a hozzájuk tartozó tárolóterület kiosztását, allokálását is elvégezve. Használatát követően a deklarált változók minden alkalmazás összes modulján belüli minden eljárás számára elérhetők lesznek.

A **Public** argumentumai a változóneveket és azok típusait, tömbváltozók esetén az indextartományokat is tartalmazó lista.

- A *változónév* argumentum a deklarálandó változónak az általános névadási konvenciók szerint képzett neve.
- A feltételes *indextartomány* argumentum a legfeljebb 60 dimenziós tömbváltozók dimenzióinak egymástól vesszővel elválasztott, [*alsóindex* **To**] *felsőindex* alakú felsorolása. Hiányzó *alsóindex* esetén annak értéke az **Option Base** utasítástól függően 0 vagy 1.
- A használható változók típusait az **As** [**New**] *típus* argumentumban írhatjuk elő. A használható típusok: **Byte**, **Boolean**, **Integer**, **Long**, **Single**, **Double**, **Currency**, **String** (változó hosszúságú karakterlánc), **String** * *hosszúság* (rögzített hosszúságú karakterlánc), **Variant**, **Boolean**, **Date**, **Object**, felhasználói típus vagy konkrét objektumtípus jelölhető meg. Minden egyes változóhoz külön-külön típusmeghatározás tartozik. Az adat- vagy objektívtípus nélkül deklarált változók — hacsak egy **Def***típus* utasítással nincs más típus előírva — alapértelmezetten **Variant** típusúak lesznek.
    - A típusdeklarációban a feltételes **New** kulcsszóval azt jelezhetjük, hogy a deklarált objektumváltozó egy Visual Basic objektum egy új példánya vagy egy külsőleg létrehozható OLE Automatizmus objektum. Ha az objektum-

változó deklarációjában nem használt a **New**, az objektumnak nem létezik példánya. A **New** kulcsszó nem használható belső adattípusú változó vagy függő OLE Automatizmus objektumok példányainak deklarálására. Az objektumváltozó csak létező, használat előtt a **Set** utasítással értéket kapott objektumra hivatkozhat, egyébként a deklarált objektumváltozó értéke **Nothing**.

Dinamikus tömb deklarálása a **Public** utasításban üres *indextartomány* megadásával történik. A deklarálás után a dinamikus tömbök dimenziójának és elemeinek száma eljárásszinten a **ReDim** utasítással definiálhatók. A **Dim**, **Private** és **Public** utasítással explicit módon létrehozott tömbváltozó újradimenzionálása hibát okoz.

A változók inicializálásakor a **Public** utasítás minden numerikus változót és tömböt nulláz, a rögzített hosszúságú karakterláncot nullákkal, a változó hosszúságút üres ("") karakterláncokkal, a **Variant** típusúakat pedig **Empty** értékekkel tölti fel. A felhasználó által definiált változótömbök az egyedi változókkal azonos módon kapnak kiinduló értéket.

A **Public** utasítás nem használható osztálymodulon belül egy rögzített hosszúságú karakteres változó deklarálására.

Mint minden eljárásszintű deklarációt, a **Public** utasítást is az eljárás elején szokás elhelyezni.

Put

Szintaxis: **Put** [#] *fájlszám*, [*rekordszám*] , *rekordváltozó*

Az utasítás egy rekordváltozót ír egy adott lemezfájlba.

- A *fájlszám* argumentum a megnyitáskor az **Open** utasításban használt fájlszámmal azonosított fájt jelöli ki a rekordváltozó írásához érvényes célfájlként.
- A *rekordváltozó* argumentum a fájlba írandó adatot tartalmazó, az objektumváltozó és tömbváltozó kivételével bármilyen típusú változót jelenthet.
- A feltételesen használható *rekordszám* argumentummal azt határozhatjuk meg, hogy az írás kezdete a **Random** módban megnyitott fájloknál melyik számú rekord, **Binary** módban megnyitott fájloknál pedig melyik számú byte legyen. A fájlok az 1-es számú rekorddal, illetve bye-tal kezdődnek. Ha a *rekordszám* hiányzik, az utána következő vesszőt akkor is ki kell tenni az írás pedig a következő (az utolsó **Get** vagy **Put** utasítással kijelölt utáni, illetve a **Seek** utasítással kijelölt) rekordnál, illetve bye-nál kezdődik.

A fájlba írás során a fájl megnyitási módjától függően néhány szempontot és szabályt kell figyelembe venni.

Random móddal megnyitott fájloknál:

- Ha az írandó *rekordváltozó* rövidebb, mint az **Open** utasítás **Len** értékeként megadott rekordhossz, a fennmaradó helyre a puffer pillanatnyi tartalmából kerülnek később már azonosíthatatlan mennyiségű, ezért a hasznos adattól szétválaszthatatlan értékek, ezért célszerű a rekordhosszat és az írandó adatmennyiséget azonosra választani.
- Az **Open** utasításban **Len** értékként megadott rekordhossznál hosszabb *rekordváltozó* hibát okoz.
- Ha a *rekordváltozó* egy változó hosszúságú karakterlánc, a **Put** egy 2 byte-os hosszadatot helyez az adatok elé, tehát az **Open** utasításban **Len** értékként megadott rekordhosszban a karakterlánc hosszánál legalább 2 byte-tal nagyobb értéket kell megadni.
- Ha a *rekordváltozó* numerikus **Variant** adat, a **Put** egy 2 byte-os, a **Variant** típuson belüli **VarType** típusazonosítót helyez az adat elé. Ennek megfelelően 2 byte azonosítja az adattípust, és a következő — az adattípustól függő számú — byte adja meg magát az adatot. **Variant** tömböt is fájlba lehet írni, azonban tömböt tartalmazó **Variant** adatot nem. Hasonlóan az előző esethez, az **Open** utasításban **Len** értékként megadott rekordhosszat a változó tárolásához szükséges hosszánál legalább 2 byte-tal nagyobbra kell megadni.
- Ha a *rekordváltozó* karakteres **Variant** adat (**VarType** 8), a **Put** egy 2 byte-os típusazonosítót és egy 2 byte-os hosszadatot helyez a karakteres adatok elé, tehát **Open** utasításban a karakterlánc aktuális hosszánál legalább 4 byte-tal nagyobb **Len** értékként megadott rekordhosszat kell megadni.
- Ha a *rekordváltozó* dinamikus tömbben található, a **Put** egy 2 byte + 8**dimenziószám* hosszúságú azonosítót helyez az adatok elé, tehát **Open** utasításban **Len** értékként megadott rekordhosszat a változók tárolásához szükséges összhossznál legalább ennyivel nagyobbra kell megadni.
- Ha a *rekordváltozó* rögzített méretű tömbben vagy — a **Variant** és a változó hosszúságú adatokat kivéve — bármely más típusú változóban található, a **Put** nem helyez azonosítót az adatok elé, tehát **Open** utasításban **Len** értékként megadott rekordhossz a változó tárolásához szükséges hosszal lehet azonos.
- A felhasználó által definiált adattípusok elemeit a **Put** egyedi adatokként írja, azonban nem tesz helykitöltő elemeket közéjük, így a **Len** értékkel megadott rekordhossz az egyedi elemek számára szükséges helyek összege lehet.
- A **Put** utasítással nem lehet objektumokat fájlba írni.

A **Binary** módban megnyitott fájloknál a **Random** módra leírtak az alábbi eltérésekkel érvényesek:

- Az **Open** utasításban a **Len** értékének nincs hatása. Mivel a **Put** folyamatosan írja az adatokat, a rekordok között nincsenek kitöltő elemek.
- A felhasználó által definiált adattípuson belüli adatok kivételével a **Put** utasítás csak az adatokat írja, nem helyez hosszadatot az adatok elé.

- A **Put** a nem felhasználó által definiált típus elemeként használt változó hosszúságú karakterláncokat a 2-byte-os hosszadat nélkül írja.

Randomize

Szintaxis: **Randomize** [*szám*]

Az utasítás a feltételes *szám* argumentumban adott bármely érvényes numerikus kifejezés értékével inicializálja az **Rnd( )** függvény véletlenszám-generátorát, vagyis beállítja az algoritmus új magértékét. Ha az utasítást argumentum nélkül használjuk, a magérték az operációs rendszer órájából a **Timer** függvénnyel kiolvasott szám lesz. A **Randomize** utasítás használata nélkül az argumentum nélküli **Rnd( )** függvény minden egyes meghíváskor ugyanazon számsorozatot állítja elő. Bővebben lásd az **Rnd( )** függvénynél leírtakat.

ReDim

Szintaxis: **ReDim** [**Preserve**] *változónév* (*indextartomány*) [**As** *típus*] [, *változónév* (*indextartomány*) [**As** *típus*]] ...

Ezzel az eljárásszinten használatos utasítással egy **Public**, **Private** vagy **Dim** utasítással előzőleg már deklarált dinamikus tömb tömbváltozóit deklarálhatjuk, illetve újradeklarálhatjuk, egyben a hozzájuk tartozó tárolóterület kiosztását, allokálását is elvégezve. A **ReDim** argumentumai a változóneveket, azok adattípusát, továbbá a tömb indextartományait tartalmazzák.

- A feltételes **Preserve** kulcsszó használatával egy létező tömb utolsó dimenziójának méretét lehet megnövelni a tömb többi adatának megőrzése mellett, így például egy négydimenziós tömbnél az első három dimenzióhoz tartozó méret és tartalom a **ReDim** utasítás végrehajtása után is változatlan marad. Ha az utolsó dimenzió méretét növeljük, annak értékei is megőrződnek, ha csökkentjük, a megszűnő dimenzió adatai elvesznek.
- A *változónév* a dimenzionálni kívánt változónak a szokásos névadási konvenciónak megfelelő nevét tartalmazza.
- A feltételes *indextartomány* argumentum a maximálisan 60 dimenziós tömbváltozók listáját tartalmazza [*alsóindex* **To**] *felsőindex* alakban, a lista elemeit egymástól vesszővel elválasztva. Ha a feltételes [*alsóindex* **To**] nincs megadva, az *alsóindex* alapértelmezett értéke az **Option Base** beállításától függően lesz 0 vagy 1.
- A használható változók típusai a feltételes **As** *típus* részben deklarálhatók, ahol **Boolean**, **Byte**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, **Single**, **String** (változó hosszúságú), **String** * *hosszúság* alakú (rögzített hosszúságú)

karakterlánc, **Variant**, felhasználó által definiált típus vagy objektum típus közül adható meg a kívánt adattípus. Minden egyes, akár azonos típusú változóhoz is külön-külön **As** *típus* meghatározás tartozik. A tömböt tartalmazó **Variant** esetén a *típus* a tömb minden egyes elemének típusát meghatározhatja, azonban a **Variant** nem módosítható más típusúra. A **Variant** tömbjének újradimenzionálás előtt a **Variant** változót explicit módon deklarálni kell.

A dinamikus, vagyis a felhasználás során meghatározható nagyságú tömbök a **Dim**, **Private** vagy **Public** utasításban üres változótömbbel deklarálhatók, majd a tömbök dimenziójának és elemeinek számát eljárásszinten ezzel a **ReDim** utasítással tetszés szerinti alkalommal (újra)definiálhatjuk. Egy adattípussal deklarált tömb típusát a **ReDim** használatával nem lehet módosítani, hacsak a tömböt nem egy **Variant** tartalmazza.

A **Dim**, **Private** és **Public** utasítással explicit módon létrehozott tömbváltozó újradimenzionálása hibát okoz.

A változók inicializálásakor a **ReDim** utasítás minden numerikus változót és tömböt nulláz, a rögzített hosszúságú karakterláncot nullákkal, a változó hosszúságút üres ("") karakterláncokkal, a **Variant** típusúakat pedig **Empty** értékekkel tölti fel. A felhasználó által definiált változótömbök az egyedi változókkal azonos módon kapnak kiinduló értéket. Az objektumra hivatkozó változót használat előtt a **Set** utasítással egy létező objektumhoz kell beállítani, egyébként értéke **Nothing**.

RegisterDatabase

Szintaxis:

Ez a — csak kompatibilitási okokból megmaradt — utasítás az **OpenDatabase** függvény használatával elérhető ODBC adatforrásnevekhez adja meg a kapcsolatleíró információkat. **DBEngine** objektumokkal inkább a **RegisterDatabase** metódus használandó.

Rem

Szintaxis 1: [*utasítások*:] **Rem** *megjegyzés*

Szintaxis 2: [*utasítások*] ' *megjegyzés*

Ez az úgynevezett nem végrehajtódó utasítás magyarázó szövegnek a programban való elhelyezésére szolgál: itt szereplő szöveg a programfutás közben nem jelenik meg a képernyőn, csak a program kilistázásával válik láthatóvá.

A **Rem** utasítás külön sorban vagy egy utasítássorban az utasítás után, azzal egy sorban, de tőle kettősponttal elválasztva is elhelyezhető (Szintaxis 1). A **Rem** helyett a Szintaxis 2 szerinti ' alakot használva, az elválasztó kettőspont szükségtelen.

Általában a megjegyzésben szereplő szöveges információk nem a felhasználónak

szólnak, szerepük elsősorban a program fejlesztője számára a felépítés áttekintését és az elkerülhetetlen hibakeresést megkönnyítő magyarázatok tárolása.

RepairDatabase

Szintaxis:

Ezzel a — csak kompatibilitási okokból megmaradt — utasítással egy sérült Microsoft Jet adatbázis helyreállítása kísérelhető meg. **DBEngine** objektumokkal inkább a **RepairDatabase** metódus használandó.

Reset

Szintaxis: **Reset**

Ez az argumentum nélküli utasítás minden aktív, az **Open** utasítással megnyitott lemezfájlt lezár, valamint az összes lemezpuffer tartalmát lemezre menti.

Resume

Szintaxis: **Resume** {[**0**] | **Next** | *sor*}

Az utasítás egy hibakezelő rutin lefutása után a program végrehajtását folytatja. Az utasítás az alábbi alakokban használható:

Ha az utasítás:	Az utasítás hatása:
Resume [**0**]	Ha a hibát okozó utasítás a hibakezelő rutinnal azonos eljárásban van, a programvégrehajtás a hibát okozó utasítással, ha pedig a hibát okozó utasítás egy meghívott, a hibakezelő rutintól eltérő eljárásban található, a hibakezelő rutint tartalmazó eljárást hívó utasítással folytatódik.
Resume Next	Ha a hibát okozó utasítás a hibakezelő rutinnal azonos eljárásban van, a programvégrehajtás a hibát okozó utasítást követő utasítással, ha pedig a hibát okozó utasítás egy meghívott, a hibakezelő rutintól eltérő eljárásban található, a hibakezelő rutint tartalmazó eljárást hívó utasítást közvetlen követő utasítással folytatódik.
Resume *sor*	A programvégrehajtás a hibakezelő rutinnal azonos eljárásban lévő, a *sor* argumentumban adott sorszámú vagy címkéjű sorban folytatódik.

Egy aktív hibakezelő rutinban a **Resume** utasítás végrehajtása előtt egy eljárás **End Sub**, **End Function** vagy **End Property** okozta befejezése hibát okoz, azonban az **Exit Sub**, **Exit Function** vagy **Exit Property** utasításon keresztüli befejezés nem.

A **Resume** utasítás hibakezelő rutinon kívüli használata hibát okoz

RmDir

Szintaxis: **RmDir** *elérési_út*

Az utasítás az *elérési_út* argumentum karakteres kifejezésével adott nevű üres könyvtár törlésére használható. Az *elérési_út* a feltételes meghajtónevet és a törlendő könyvtárnevet tartalmazza:

[*meghajtónév*:] [\ ] *könyvtárnév*

Ha az argumentumból hiányzik a *meghajtónév*, a könyvtár az éppen aktív meghajtóról törlődik. Fájlt tartalmazó könyvtárat a **Kill** utasítással lehet törölni, ilyen esetben az **RmDir** használata hibát okoz.

Rollback

Szintaxis:

Ez az argumentum nélküli — csak kompatibilitási okokból megmaradt — utasítás lezárja az egyetlen egységként tekintett adatbázisokban végzett módosítások sorozatát, valamint az adatbázist (vagy adatbázisokat) az aktuális tranzakciót megelőző állapotba állítja vissza. **Workspace** objektumokkal a tranzakció lezárására és az adatbázis visszaállítására inkább a **Rollback** metódus használandó.

RSet

Szintaxis: **LSet** *karakteres_változó* = *karakteres_kifejezés*

Az **LSet** utasítás egy karakteres kifejezést egy változón belül balra igazítva helyez el. Az argumentumok a következők:

- A *karakteres_változó* argumentum karakteres kifejezése a változó nevét adja meg.
- A *karakteres_kifejezés* argumentumban a balra igazítottan elhelyezendő karakteres kifejezést tartalmazza.

Az **RSet** utasítást nem lehet felhasználó által definiált típusokkal használni, azonban az egyéb tudnivalók értelemszerűen azonosak az **LSet** utasításnál leírtakkal.

SavePicture

Szintaxis: **SavePicture** *képtulajdonság*, *karakteres_kifejezés*

Az utasítással egy **Form** objektum, **PictureBox** vezérlőelem vagy **Image** vezérlőelem grafikáját fájlba lehet menteni.

- A *képtulajdonság* argumentumban az a **Picture** vagy **Image** tulajdonságot kell megadni, amelyből a képfájlt létre lehet hozni.
- A *karakteres_kifejezés* argumentum karakteres kifejezésébe annak a fájlnak a nevét kell megadni, amelybe a grafikát menteni akarjuk.

Ha akár a programfejlesztés, akár a programfutás során egy képet fájlból töltünk be a **Picture** tulajdonságba, a mentés csak az eredeti képformátumban történhet. Az **Image** tulajdonságból a kép mindig bitképes (.BMP) formában tárolódik.

SaveSetting

Szintaxis: SaveSetting (***appname***, ***section***, ***key***, ***setting***)

Az utasítás menti vagy létrehozza egy alkalmazás Windows regisztrációs bejegyzését. A **SaveSetting** utasítás névvel hivatkozott argumentumai a következők:

- Az ***appname*** argumentum karakteres kifejezésében kell megadni annak az alkalmazásnak vagy projektnek a nevét, amelyhez a beállítás tartozik.
- A ***section*** argumentum karakteres kifejezésben annak a szakasznak a nevét kell megadni, amelybe a kulcs beállítását tárolni akarjuk.
- A ***key*** argumentum karakteres kifejezésében a tárolandó kulcs nevének kell szerepelnie.
- Végül, a ***setting*** argumentum a ***key*** beállítási értékét tartalmazó kifejezés.

Ha a beállított kulcsérték bármely okból nem menthető, hibajelzést kapunk.

Seek

Szintaxis: **Seek** [#] *fájlszám*, *pozíció*

Az utasítással egy megnyitott fájl következő írási vagy olvasási műveletének pozíciója állítható be.

- A *fájlszám* argumentumban az **Open** utasításban a fájl megnyitására használt számmal azonos érték megadásával lehet a pozícionálandó fájt kijelölni.
- A *pozíció* argumentum a következő írási/olvasási pozíciót megadó, 1 és 2146483647 közötti szám. Nulla vagy negatív *pozíció* hibát okoz.

A **Seek** utasításnak a fájlvég utánra történő végrehajtását követő írási művelet megnöveli a fájl méretét.

A **Get** és **Put** utasítások felülírják a **Seek** utasítás által beállított pozícióértéket.

Select Case

Szintaxis:

Select Case *vizsgálandó_kifejezés*
[**Case** *kifejezéslista*
[*utasításcsoport1*]]
...
[**Case Else**
[*utasításcsoport_n*]]
End Select

A **Select Case** feltételes utasítás egy kifejezés értékétől függően több utasításcsoport közül egyet hajt végre. Az utasítás argumentumai a következők:

- A *vizsgálandó_kifejezés* bármely érvényes numerikus vagy karakteres kifejezés lehet.
- A *kifejezéslista* argumentum a következő alakú kifejezés vagy kifejezések vesszővel elválasztott listája lehet: *kifejezés*, *kifejezés1* **To** *kifejezés2* vagy **Is** *összehasonlítási_operátor kifejezés*.
  - Az értéktartományt kijelölő *kifejezés1* **To** *kifejezés2* alakban a *kifejezés1* értékének kisebbnek kell lennie a *kifejezés2* értékénél, például -8 **To** -2, vagy alma **To** bab. Karakteres kifejezések esetén az összehasonlítás a karakterkódok szerint történik, tehát alma < Alma < ALMA.
  - Az ugyancsak értéktartományt kijelölő **Is** *összehasonlítási_operátor kifejezés* alakban az *összehasonlítási_operátor* az **Is** és **Like** kivételével bármelyik lehet. Ha az értéktartomány kijelölésében használt **Is** kulcsszó hiányzik, azt a Visual Basic automatikusan elhelyezi. Az **Is** kulcsszó nem azonos az **Is** összehasonlítási operátorral!
- Az *utasításcsoport1*, ... *utasításcsoport_n* argumentumok azokat az utasításokat vagy utasítások csoportjait tartalmazzák, amelyek közül egyet kell végrehajtani a *vizsgálandó_kifejezés* és a *kifejezéslista* értékétől függően.

A **Select Case** utasítás úgy működik, hogy a *vizsgálandó_kifejezés* az argumentum aktuális értékét egymás után összehasonlítja a *kifejezéslista* tartalmával. Ha egyezést talál, végrehajtja a hozzá tartozó utasításcsoport utasításait, majd a program futása az **End Select** utáni sorban folytatódik. Megengedett a kifejezéslistában ugyanazon elem többszöri használata is, azonban csak a sorrendben első egyezéshez tartozó utasítások hajtódnak végre.

Ha például a *vizsgálandó_kifejezés* előre nem látható értéke miatt nincs egyezés, a **Case Else** utáni *utasításcsoport_n* — általában hibajelzés — végrehajtása következik be.

Az **End Select** az utasítás lényegi, kötelezően alkalmazandó része, azonban a **Case**

Else rész el is hagyható, így egyezés hiányában a programfutás azonnal az **End Select** utáni sorban fog folytatódni.

Több **Select Case** is egymásba ágyazható, de minden egyes **Select Case** utasításnak kell lennie **End Select** lezárásának.

SendKeys

Szintaxis: **SendKeys** ***string*** [, ***wait***]

Az utasítással a billentyűzetről való beíráshoz hasonlóan lehet az aktív ablak vagy vezérlőelem számára egy vagy több billentyűleütésből álló üzenetet küldeni. Nem Windows alatt futó alkalmazási programok számára a **SendKeys** utasítással nem lehet billentyűleütéseket szimulálni.

- A névvel hivatkozott ***string*** argumentum a billentyűleütéseket tartalmazó karakteres kifejezés.
- Az utasítás a feltételes ***wait*** argumentumban adott **Boolean** értékkel egészíthető ki, amely a várakozási módot írja elő. Ha az érték **True**, a vezérlés az eljárásra történő visszaadása előtt várakozik a billentyűnyomás feldolgozására, az alapértelmezett **False** értékénél pedig a vezérlés azonnal visszatér az eljárásra.

A ***string*** argumentumban használható betű- és számkaraktereket a hozzájuk tartozó billentyűk lenyomásával lehet beírni.

A képernyőn nem megjelenő — elsősorban vezérlő billentyűk — beírásához ugyancsak kapcsos zárójelek közé tett kódok használhatók a következő lista szerint:

Ha a billentyű:	A ***string***:	Ha a billentyű:	A ***string***:
BREAK	{BREAK}	F1	{F1}
CAPS LOCK	{CAPSLOCK}	F2	{F2}
HELP	{HELP}	F3	{F3}
TAB	{TAB}	F4	{F4}
FEL	{UP}	F5	{F5}
LE	{DOWN}	F6	{F6}
JOBBRA	{RIGHT}	F7	{F7}
BALRA	{LEFT}	F8	{F8}
PG DN	{PGDN}	F9	{F9}
PG UP	{PGUP}	F10	{F10}
HOME	{HOME}	F11	{F11}
END	{END}	F12	{F12}
NUM LOCK	{NUMLOCK}	F13	{F13}
PRTSC	{PRTSC}	F14	{F14}
ESC	{ESCAPE}	F15	{F15}
	{ESC}		

INS	{INSERT}	F16	{F16}
DEL	{DELETE} {DEL}	BACKSPACE	{BACKSPACE} {BKSP} {BS}
SCROLL LOCK	{SCROLLOCK}	ENTER	{ENTER}

A "+", "^", "~ " és "%" karaktereknek a **SendKeys** utasításban különleges jelentése van:

Ha a billentyű:	A *string*:
ALT	%
CTRL	^
ENTER	~
SHIFT	+

Ezek a karakterek csak kapcsos zárójelben jelentik saját magukat, például {%}. Magát a kapcsos zárójelet a {{} és {}} alakban lehet megadni.

A **SendKeys** utasítás *string* argumentumában például a CTRL billentyű lenyomva tartása mellett az A, B és C billentyűk leütését a ^(ABC) kifejezés, a CTRL billentyű lenyomva tartása mellett az A, majd a CTRL billentyű lenyomva tartása nélkül a B és C billentyűk leütését a ^A(BC) kifejezés jelöli.

Egy billentyű egymás utáni ismételt lenyomását az egyszerűsített formában is írhatjuk, például a PgDn billentyű tizenháromszori lenyomásának a következő kifejezés felel meg: {PGDN 13}.

Egyes alkalmazások számára nem lehet a PRTSC billentyűleütést kiküldeni.

Set

Szintaxis: **Set** *objektumváltozó* = { [**New**] *objektumkifejezés* | **Nothing** }

Az utasítás egy objektumhivatkozást állít be egy változóba vagy tulajdonságba. Az egyes argumentumok értelmezése a következő:

- Az *objektumváltozó* argumentum egy objektum vagy tulajdonság névadási konvencióknak megfelelő neve. Az objektumváltozó típusának összhangban kell állnia azzal az objektummal, amelyhez beállítjuk.
- Az *objektumkifejezés* argumentum egy objektum nevét tartalmazó kifejezés, ugyanazon objektumtípus egy másik deklarált változója vagy egy ugyanazon objektumtípus egy objektumát visszaadó függvény vagy metódus lehet. A feltételes **New** kulcsszó a jelöli, hogy egy Visual Basic objektum vagy külsőleg létrehozható OLE Automatizmus objektum új példányát hozzuk létre.
- Az *objektumváltozó* argumentumnak a **Nothing** értéket adva az objektumváltozó és bármely objektum között fennálló kapcsolat megszakad. Mivel az előzőleg hi-

vatkozott objektumra már nincs hivatkozó változó, a vele kapcsolatos rendszer- és memória-erőforrások is felszabadulnak.

A **Dim**, **Private**, **Public**, **ReDim** és **Static** utasítások csak az objektumra hivatkozó változó deklarálását végzik el. Az aktuális objektumhoz történő hivatkozást a **Set** utasítással kell beállítani, azonban ilyenkor a változóhoz nem jön létre objektummásolat. Ugyanarra az objektumra több objektumváltozó is hivatkozhat. Mivel ezek csak hivatkozások, az objektumot érintő minden változás a rá hivatkozó minden változóra vagy változókra is kihat.

A **Set** utasításban a **New** záradékot használva az objektumnak egy tényleges másolata jön létre. Ez az új példány az első rá vagy az objektumok tagjainak egyikére vonatkozó hivatkozásig nem töltődik be.

SetAttr

Szintaxis: SetAttr ***pathname***, ***attributes***

Az utasítással egy fájlattribútum beállítása végezhető el.

A fájl nevét, továbbá a meghajtó és a könyvtár megnevezését a névvel hivatkozott ***pathname*** argumentum karakteres kifejezése tartalmazza.

Az ***attributes*** argumentum numerikus érték vagy szimbolikus konstans is lehetnek a következő táblázatnak megfelelően:

Ha a szimbolikus konstans:	Értéke:	Értelmezése:
vbNormal	0	Normál fájl
vbReadOnly	1	Csak olvasható fájl
vbHidden	2	Rejtett fájl
vbSystem	4	Rendszer fájl
vbArchive	32	Az utolsó biztonsági másolat készítése óta módosított fájl

A **SetAttr** utasítás megnyitott fájlra alkalmazása hibát okoz.

SetDataAccessOption

Szintaxis:

A Microsoft Jet adatbázis-szerkesztő inicializálási fájl helyét megadó utasítás csak kompatibilitási okból maradt meg, helyette a **DBEngine** objektum **IniPath** metódusát lehet használni.

SetDefaultWorkspace

Szintaxis:

Ez — a védett hozzáférésű Microsoft Jet adatbázis számára felhasználói azonosítónevet (ID) és jelszót megadó — utasítás csak kompatibilitási okokból maradt meg, helyette ugyanerre a célra a **Workspace** objektum tulajdonságait lehet használni.

Static

Szintaxis: **Static** *változónév* [([*indextartományok*])] [**As** [**New**] *adattípus*] [, *változónév* [([*indextartományok*])] [**As** [**New**] *adattípus*]] ...

Az eljárásszinten használt **Static** utasítással az eljárás futása alatt értéküket megőrző változók deklarálása és a hozzájuk tartozó tárolóterület allokálása végezhető el. Az utasítás argumentumai a deklarálandó változóneveket és azok típusait, továbbá a dimenzionált tömbök indextartományait sorolják fel a következőképpen:

- A *változónév* a változónak a szokásos névadási konvenciónak megfelelő nevét tartalmazza.
- A feltételes *indextartomány* argumentum a maximálisan 60 dimenziós tömbváltozók listáját tartalmazza [*alsóindex* **To**] *felsőindex* alakban, a lista elemeit egymástól vesszővel elválasztva. Ha a feltételes [*alsóindex* **To**] nincs megadva, az *alsóindex* alapértelmezett értéke az **Option Base** beállításától függően lesz 0 vagy 1.
- A használható változók típusai a feltételes **As** [**New**] *típus* részben deklarálhatók, ahol **Boolean**, **Byte**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, **Single**, **String** (változó hosszúságú), **String** * *hosszúság* alakú (rögzített hosszúságú) karakterlánc, **Variant**, felhasználó által definiált típus vagy objektum típus közül adható meg a kívánt adattípus. Minden egyes, akár azonos típusú változóhoz is külön-külön **As** *típus* meghatározás tartozik. Az adat- vagy objektívtípus nélkül deklarált változók — hacsak egy **Def***típus* utasítással nincs más típus előírva — alapértelmezetten **Variant** típusúak lesznek.
    - A típusdeklarációban a feltételes **New** kulcsszóval azt jelezhetjük, hogy a deklarált objektumváltozó egy Visual Basic objektum egy új példánya vagy egy külsőleg létrehozható OLE Automatizmus objektum. Ha objektumváltozó deklarációban nem használt a **New**, az objektumnak nem létezik példánya. A **New** kulcsszó nem használható belső adattípusú változó vagy függő OLE Automatizmus objektumok példányainak deklarálására. Az objektumváltozó csak létező, használat előtt a **Set** utasítással értéket kapott objektumra hivatkozhat, egyébként a deklarált objektumváltozó értéke **Nothing**.

Egy modul programkódjának futásakor a **Static** utasítással deklarált változók az értéküket a modul alapállapotba állításáig vagy újra futtatásáig megőrzik, így a nem **Static** deklarálású eljárások egyes változói ilyen explicit módon tehetők **Static** tulajdonságúvá.

Az utasítás az eljáráson belüli **Static** változók adattípusának, illetve objektumtípusának deklarációjára is használható.

A **Static** utasításnak és a **Static** kulcsszónak a változók élettartamára eltérő hatása van: a **Static** kulcsszóval deklarált eljárás esetén az eljáráson belüli összes helyi változó tárolóterülete egyszer kerül kiosztásra, a változók értéke pedig a teljes program futása alatt megőrződik, ezzel szemben a nem **Static** deklarálású eljárások **Static** változói számára a tárolóterület az eljárás minden egyes meghívásakor kiosztásra kerül, az eljárás befejeződése után pedig felszabadul.

A változók inicializálásakor a **Static** utasítás minden numerikus változót és tömböt nulláz, a rögzített hosszúságú karakterláncot nullákkal, a változó hosszúságút üres ("") karakterláncokkal, a **Variant** típusúakat pedig **Empty** értékekkel tölti fel. A felhasználó által definiált változótömbök az egyedi változókkal azonos módon kapnak kiinduló értéket.

Mint minden eljárásszintű deklarációt, a **Static** utasítást is az eljárás elején szokás elhelyezni.

Stop

Szintaxis: **Stop**

Ez a programban bárhová helyezhető, a töréspontho hasonló hatású argumentum nélküli utasítás felfüggeszti a program futását.

A **Stop** az **End** utasítástól eltérően — a futtatható .EXE fájlok kivételével — nem zárja le a fájlokat és nem törli a változókat, így a Run menü Continue vagy a Step Into, Step Over parancsával a programfutás folytatható.

Sub

Szintaxis:

[Private | Public] [Static] Sub *eljárásnév* [(*argumentumlista*)]
    [*utasításcsoport1*]
    [**Exit Sub**]
    [*utasításcsoport2*]
End Sub

A **Function** eljáráshoz hasonló felépítésű **Sub** a Visual Basic egyik lényeges építőköve, hiszen az alkalmazási program minden végrehajtandó utasítását vagy **Sub** vagy **Function** eljárásban kell elhelyezni. A kétféle eljárás között lényegi eltérést jelent, hogy a **Function** mindig egy értéket ad vissza, tehát függvényként használható, a **Sub** azon-

ban nem használható kifejezésekben, használatával csak utasítások hajthatók végre és változók értékei módosíthatók.

A **Sub** utasítás az eljárás nevét, argumentumait és programkódját a következők szerint deklarálja:

- Az utasításban feltételesen használható **Private** vagy **Public** záradékkal a **Sub** eljárás hatóköre írható elő: az alapértelmezett **Private** deklarációval az eljárás használata csak a deklarációt tartalmazó modulra korlátozódik, míg a **Public** deklarációval minden modul összes eljárásai számára is elérhető lesz.
- Az ugyancsak feltételes **Static** záradék azt szabja meg, hogy a **Sub** eljárás helyi változóinak értékei a hívások között is megtartsák-e értéküket. Az eljárásban használt nem helyi változókra a **Static** hatástalan.
- A **Sub** eljárásnak a változónevekre érvényes konvencióknak megfelelően képzett nevét az *eljárásnév* argumentumban kell megadni.
- Az eljárás nevét az eljárásban használt, annak meghívásakor átadandó változó a következő, [**Optional**] [**ByVal** | **ByRef**] [**ParamArray**] *változónév* [( )] [**As** *típus*] általánosított alakú, több változó esetén azok vesszővel elválasztott, zárójelbe tett listája, az *argumentumlista* követi.
  - A feltételes **Optional** azt jelzi, hogy az ezt követő argumentum feltételes, azaz az adott eljárás meghívásakor nem kötelező megadni. Ha egy argumentum feltételes, az *argumentumlista* ezt követő további argumentumainak is feltételeseknek, azaz az **Optional** kulcsszóval deklaráltaknak és **Variant** típusúaknak kell lenniük. Az **Optional** és a **ParamArray** kulcsszavak egy argumentumon belül nem használhatók.
  - A **ByVal** vagy **ByRef** feltételes kulcsszó a változóértékek külső eljárás számára történő átadási módját határozza meg. A **ByVal** esetén a numerikus adattípusok éppen aktuális értéke átadás előtt a **Declare** által meghatározott adattípussá konvertálódik, míg a **String** típusú változó esetében a nullával lezárt karakterlánc címe adódik át az eljárásnak. A **ByVal** használata nélkül a **String** típusú változó karaktersorozat-leírásként kerül át a hívott **Sub** eljárásba. A **ByVal** ellentéte a **ByRef**, ami hatására az azt követő argumentum referenciával (címmel) adódik át a meghívott eljárásnak.
  - Az ugyancsak feltételes **ParamArray** csak az argumentumlista utolsó eleme előtt állhat, jelezve, hogy az adott argumentum egy **Variant** elemekből álló **Optional** tömb. Ilyen módon tetszőleges számú és típusú további argumentumot adhatunk át egy eljárásnak, természetesen csak akkor, ha az felkészült ezek fogatására. A **ParamArray** kulcsszóval együtt a **ByVal**, **ByRef** és **Optional** kulcsszavak egyike sem szerepelhet.
  - A lista egyetlen kötelező eleme a külső eljárásnak átadandó, a változónevek megadására vonatkozó konvencióknak megfelelő *változónév* argumentum. Az eljárásban nem expliciten deklarált változók használata esetén névütkö-

zési problémák következhetnek be, ezért a programkódban célszerű az **Option Explicit** utasítással kötelező deklarálást előírni.

- Az eljárásnak átadandó változó típusát a feltételes **As** *típus* adja meg. A használható adattípus **Boolean**, **Byte**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, kizárólag változó hosszúságú **String**, **Variant**, felhasználói által definiált vagy egy objektumtípus lehet.

- A **Sub** eljárásban végrehajtandó utasításokat a feltételes *utasításcsoport1* és *utasításcsoport2* tartalmazza.
- A tetszőleges számban elhelyezhető **Exit Sub** kulcsszóval lehet a **Sub** eljárásból kilépni. Ekkor a programfutás az eljárást meghívó utasítást követő utasítás végrehajtásával folytatódik.
- A **Sub** eljárást az **End Sub** zárja le.

A **Sub** eljárást az *eljárásnév* és az azt követő (*argumentumlista*), azaz a zárójelben felsorolt argumentumok megadásával lehet meghívni.

A **Sub** elsőként az *utasításcsoport1* utasítást vagy utasításcsoportot (általában valamilyen vizsgálatot) hajtja végre, majd annak eredményétől függően az *utasításcsoport2* utasítás vagy utasításcsoport végrehajtását kezdi meg, vagy az **Exit Sub** parancson keresztül azonnal a **Sub** eljárást hívó utasítás utáni utasításra lép.

A **Sub** eljárások rekurzívak, azaz saját magukat is hívhatják, de gondoskodni kell arról, hogy a rekurzió ne legyen végtelen. A rekurzívan használandó eljárások deklarálásakor ne használjuk a **Static** kulcsszót.

A **Sub** számára átadandó változók közül az eljárásra nézve mindig lokálisak az explicit módon — például a **Dim** használatával — deklaráltak, a nem közvetlenül deklaráltak közül pedig azok, amelyek az eljáráson kívül, magasabb szinten nem kerültek közvetlenül deklarálásra. Az eljárásokban nem közvetlenül deklarált változók is használhatók, de ilyenkor fennáll a névütközés veszélye, vagyis ha az eljárás olyan nem deklarált változóra hivatkozik, amely neve egy másik eljárás, konstans vagy változó nevével azonos, azt az eljárás modulszintűnek tekinti. A névütközés az **Option Explicit** utasítás használatával kerülhető el.

Sub eljárást nem definiálhatunk másik **Function**, **Sub** vagy **Property** eljárásokon belül.

Time

Szintaxis: **Time** = *idő*

Az utasítás a rendszeren belüli idő beállítására használható. Az időpontot tartalmazó *idő* argumentum lehet időpontként értelmezhető karakteres, numerikus vagy ezek kombinációját tartalmazó kifejezés.

Ha az *idő* karakteres kifejezés, a **Time** utasítás azt a rendszerben érvényes időtagoló karakterek alapján kísérli meg értelmezni. Az értelmezhetetlen kifejezés hibát okoz.

Az utasítás az MS-DOS 3.3-s előtti verziókban nem módosítja a számítógép CMOS RAM-jában tárolt dátumot, vagyis a beállított dátum csak a számítógép kikapcsolásáig marad érvényes. Ilyen rendszerekben a dátum tartós módosítása csak a rendszer dokumentációjában leírt módon lehetséges.

Type

Szintaxis:

[Private | Public] Type *változónév*
elemnév1 [(*indextartomány*)] **As** *adattípus*
[*elemnév2* [(*indextartomány*)] **As** *adattípus*]
...
End Type

Ezzel a modulszinten használt utasítással egy vagy több elemet tartalmazó, felhasználó által definiált változótípus deklarálható. Az utasításban használt argumentumok a következők:

- Az utasításban feltételesen használható **Private** vagy **Public** záradékkal a felhasználó által definiált változótípus hatóköre írható elő. A **Private** deklarációval a típus használata csak a deklarációt tartalmazó modulra korlátozható, míg a **Public** deklarációval minden modul összes eljárásai számára is elérhető lesz. A standard modulokban a **Public** az alapértelmezett hatókör, míg az osztálymodulokban ez módosíthatatlanul **Private**.
- A *változónév* a deklarálni kívánt, felhasználó által definiált változótípus szokásos névadási konvenciónak megfelelő nevét tartalmazza.
- Az *elemnév1*, *elemnév2*, ... argumentumok a felhasználó által definiált változótípus elemeinek nevét tartalmazzák. A nevek a szokásos névadási konvenciónak megfelelők lehetnek, azonban itt névként a kulcsszavak használata is megengedett.
- A feltételes *indextartomány* argumentum a maximálisan 60 dimenziós tömbváltozók listáját tartalmazza [*alsóindex* **To**] *felsőindex* alakban, a lista elemeit egymástól vesszővel elválasztva. Ha a feltételes [*alsóindex* **To**] nincs megadva, az *alsóindex* alapértelmezett értéke az **Option Base** beállításától függően lesz 0 vagy 1.
- Az elemek adattípusa az **As** *típus* részben deklarálhatók, ahol **Boolean**, **Byte**, **Currency**, **Date**, **Double**, **Integer**, **Long**, **Object**, **Single**, **String** (változó hosszúságú), **String** * *hosszúság* alakú (rögzített hosszúságú) karakterlánc, **Variant**, felhasználó által definiált másik típus vagy objektum típus használható. Minden egyes változóhoz külön-külön **As** *típus* meghatározás tartozik.

A **Type** utasítással történt típusdeklarációt követően a deklaráció hatókörében a

Dim, **Private**, **Public**, **ReDim** vagy **Static** utasítással bárhol deklarálhatunk az adott típusba tartozó változót.

A **Type...End Type** blokkok nem tartalmazhatnak sorszámot vagy címkét.

A felhasználó által definiált típuson belül deklarált rögzített méretű tömbök dimenzióját változó helyett numerikus számértékkel vagy állandóval kell megadni.

A felhasználó által definiált típuson belüli tömbök legkisebb indexét az **Option Base** beállítása határozza meg.

Unload

Szintaxis: **Unload** *objektum*

Az utasítás egy űrlapot vagy annak valamely vezérlőelemét törli a memóriából. A törlendő **Form** objektum vagy vezérlőelem-tömb nevét az utasítás *objektum* azonosítójában kell megadni.

Az űrlap vagy vezérlőelem törlése akkor szükséges vagy célszerű, ha a memóriát más célokra akarjuk felhasználni vagy a tulajdonságokat a kiinduló értékeikre akarjuk visszaállítani.

Az űrlap törlésekor elsőként egy **Query_Unload**, majd egy **Form_Unload** esemény következik be. Ha a két esemény közül bármelyik **Cancel** argumentuma **True** beállítású, az űrlap nem törlődik.

Egy **MDIForm** objektum törlésekor elsőként az **MDIForm** objektumra vonatkozó **Query_Unload** esemény következik be, ez követi a minden egyes MDI-alablakra vonatkozó **Query_Unload** és **Form_Unload** esemény, végül pedig az **MDIForm** objektumra vonatkozó **Form_Unload** esemény következik be.

A törlést követően az adott űrlapon a programfutás alatt elhelyezett vezérlőelemek továbbra már nem lesznek elérhetők, a programfejlesztési fázisban elhelyezett vezérlőelemek azonban érintetlenül megmaradnak, noha az újbóli betöltés során egyes, a programfutás során rajtuk vagy tulajdonságaikon végrehajtott változtatások elvesznek. Ugyancsak elvesznek az űrlap tulajdonságainak változásai is.

Az űrlap törlésekor csak a megjelenített összetevők törlődnek, az űrlapmodullal kapcsolatos programkód a memóriában marad.

A vezérlőelemek tömbjéből csak a programfutás alatt csatolt elemek törölhetők. Az eltávolított vezérlőelemek tulajdonságai azok ismételt betöltéskor újrainicializálódnak.

Unlock

Szintaxis:

Az utasítás használatára vonatkozó tudnivalók a **Lock...Unlock** utasításnál szerepelnek.

While...Wend

Szintaxis:

While *feltétel*
[*utasításcsoport*]
Wend

Ez az utasítás egy utasítást vagy utasításcsoportot hajt végre mindaddig ismétlődően, amíg egy feltétel **True**.

- A *feltétel* argumentumban olyan numerikus vagy karakteres kifejezésnek kell szerepelnie, amelyet kiértékelve **True** vagy **False** eredményt kapunk.
- A feltételes *utasításcsoport* egy vagy több utasítást foglalhat magában, amelyek végrehajtása a *feltétel* teljesülésétől függően következik be.

Ha a *feltétel* **True**, az utasítás végrehajtja az *utasításcsoport* parancsait, majd a program ismét visszatér a *feltétel* kiértékeléséhez. A ciklus mindaddig folytatódik, míg a kiértékelés eredménye **False** nem lesz — ekkor a program végrehajtása a **Wend** sor után következő utasításra kerül.

A **While...Wend** utasítások tetszőlegesen egymásba ágyazhatók. Minden egyes **While** párja a következő **Wend**.

A **While...Wend** utasításnál strukturáltabb és rugalmasabb programozási módot tesz lehetővé a **Do...Loop** utasítás.

Width

Szintaxis: **Width #** *fájlszám*, *szélesség*

Az utasítással egy, az **Open** utasítással megnyitott fájl kimeneti sorszélessége állítható be.

- A *fájlszám* argumentum a fájlt az **Open** megnyitására használt fájlszám alapján jelöli ki.
- A *szélesség* egy 0 és 255 közötti értékű numerikus kifejezés, amely azt jelöli meg, hogy egy sorban hány karakter jelenhet meg az új sor megkezdése előtt. Az alapértelmezett 0 érték esetén a sorszélesség nem korlátozott.

With

Szintaxis:

With *objektum*
[*utasításcsoport*]
End With

A **With** utasítás egy utasítást vagy utasításcsoportot hajt végre egy objektumon vagy felhasználó által definiált típuson.

- Az *objektum* azonosítója egy objektum vagy felhasználó által definiált típus neve. Egy **With** utasításon belül nem lehet módosítani az objektum nevét.
- A feltételes *utasításcsoport* egy vagy több utasítást foglalhat magában, amelyeket az objektumon akarunk végrehajtani.

Az utasítással az objektum nevének ismételt megadása nélkül tudunk egy vagy több utasítást végrehajtani. Egy **With** utasítást csak egyetlen objektumon lehet használni.

A **With** utasítások tetszőlegesen egymásba ágyazhatók. Mivel a külső **With** blokkok a belső blokkokon belül maszkoltak, egy belső **With** blokkban teljesen meghatározott objektumhivatkozást kell használni egy külső **With** blokk objektumeleméhez. Nem szabad a **With** blokkba beleugrani, vagy abból kiugrani, mivel az *utasításcsoport* végrehajtása hibát vagy előre nem látható eredmény okoz a **With** vagy **End With** utasítás végrehajtása nélkül.

Write#

Szintaxis: **Write#** *fájlszám*, [*kimeneti_lista*]

Az utasítással egy soros (szekvenciális) fájlba nyers, formázatlan adatok írhatók.

- A *fájlszám* argumentum a fájlt az **Open** megnyitására használt fájlszám alapján jelöli ki.
- A feltételes *kimeneti_lista* argumentum a fájlba írandó, egy vagy több, vesszővel, pontosvesszővel vagy szóközzel tagolt numerikus vagy karakteres kifejezés. Ha ez az argumentum hiányzik, azonban a *fájlszám* argumentum után vessző szerepel, a fájlba egy üres sor kerül.

Adatok fájlba írásához a **Write#** utasítást használva számos általános feltételt kell figyelembe venni ahhoz, hogy a beírt adatok az **Input#** utasítással a területi beállítástól függetlenül mindig helyesen legyenek beolvashatók és értelmezhetők:

- A numerikus adatokban tizedesjelként mindig tizedespontot (.) kell használni.
- A **Boolean** adatok esetén #TRUE# vagy #FALSE# kerül kiírásra. A **True** vagy **False** kulcsszavak a területi beállítástól függetlenül, fordítás nélkül kerülnek a fájlba.
- A **Date** adatok univerzális formában kerülnek kiírásra. Ha akár a dátum, akár az időpont összetevő hiányzik, csak a megadott rész kerül kiírásra.
- Ha a *kimeneti_lista* **Empty**, a **Write#** semmit sem ír a fájlba, de a **Null** adat #NULL# alakban kerül kiírásra.
- Az **Error** adatok #ERROR *hibaszám*# alakban kerülnek a fájlba. Az **Error** kulcsszó a területi beállítástól függetlenül, fordítás nélkül kerül kiírásra.

Eltérően a **Print#** utasítástól, a **Write#** a fájlba írt adatok közé elválasztó karaktereket helyez: az egyes tételek közé vesszőt, a karaktersorozatok elé és mögé pedig idézőjelet. A lista utolsóként írt karaktere után újsor karakter, azaz **Chr**(13) vagy kocsivisszasoremelés karakter, azaz **Chr**(13) + **Chr**(10) kerül a fájlba.

Metódusok

Add

Szintaxis: *objektum*.**Add** (***item*** [, ***key***] [, ***before***] [, ***after***])

Ez a **Collection** objektumokon használható metódus egy tagot ad az objektumhoz. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A névvel hivatkozott ***item*** argumentum bármely típusú, az objektumgyűjteményhez adandó tagot meghatározó kifejezés lehet.
- A feltételes, névvel hivatkozott ***key*** argumentum egy egyedi karakteres kifejezés, amely az új taghoz történő hozzáféréshez a pozícióindex helyett használható karakteres kulcsot adja meg. Az objektumgyűjtemény valamely tagjához tartozó karakteres kulccsal azonos **key** argumentum hibát okoz.
- A feltételes, névvel hivatkozott ***before*** argumentum az objektumgyűjteményen belüli relatív pozíciót megadó kifejezés. Az objektumgyűjteményhez hozzáadandó új tag a ***before*** argumentumban meghatározott tag elé kerül. Ha a ***before*** numerikus kifejezés, annak értéke 1 és az objektumgyűjtemény **Count** tulajdonságának értéke között lehet, míg karakteres kifejezés esetén az objektumgyűjtemény egyes elemeit a halmazhoz adásukkor használt ***key*** argumentumukkal azonosan kell megadni. Természetesen a ***before*** és az ***after*** argumentumok egyidőben nem használhatók, és az objektumgyűjtemény nem létező tagjára történő hivatkozás hibát okoz.
- A feltételes, névvel hivatkozott ***after*** argumentum az objektumgyűjteményen belüli relatív pozíciót megadó kifejezés. Az objektumgyűjteményhez hozzáadandó új tag az ***after*** argumentumban meghatározott tag után kerül. Ha az ***after*** numerikus kifejezés, annak értéke 1 és az objektumgyűjtemény **Count** tulajdonságának értéke között lehet, míg karakteres kifejezés esetén az objektumgyűjtemény egyes elemeit a halmazhoz adásukkor használt ***key*** argumentumukkal azonosan kell megadni. Természetesen a ***before*** és az ***after*** argumentumok egyidőben nem használhatók, és az objektumgyűjtemény nem létező tagjára történő hivatkozás hibát okoz.

AddItem

Szintaxis: *objektum*.**AddItem** *elem* [, *index*]

Ez a metódus egy új elemet ad egy **ComboBox** vagy **ListBox** vezérlőelemhez, illetve egy új sort ad egy **Grid** vezérlőelemhez. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *elem* argumentum az objektumgyűjteményhez adandó tagot meghatározó karakteres kifejezés. Kizárólag a **Grid** vezérlőelemhez a tabuláló karakter, azaz **Chr$**(09) használható egy újonnan csatolt sor minden egyes oszlopába beillesztendő többszörös kifejezések elválasztására.
- A feltételes *index* argumentum egy új elem vagy sor objektumon belüli pozícióját meghatározó egész szám. Egy **ListBox** vagy **ComboBox** első elemének, valamint egy **Grid** első sorának esetén az argumentum értéke 0. Ha az *index* argumentum nincs megadva, az *elem* a **Sorted** tulajdonságának értékétől függően vagy a megfelelő rendezési sorba (**True**), vagy a lista végére (**False**) kerül.

A **Data** vezérlőelemhez kötött **ListBox** vagy **ComboBox** vezérlőelemek nem támogatják ezt a metódust.

AddNew

Szintaxis: *rekordhalmaz*.**AddNew**

Ez a metódus egy új, szerkeszthető rekordot hoz létre a *rekordhalmaz* azonosítójában megnevezett, nyitott eredményhalmaz és tábla típusú **Recordset** (továbbá az elavult **Dynaset** vagy **Table**) objektumokon.

A metódus minden mezőt a tábla típusú **Recordset** objektum esetén alapértelmezettként beállított **Null** értékekkel, vagy ha más alapértelmezett érték az érvényes, akkor azokkal tölti fel. Az új, módosított rekordot az **Update** metódussal lehet menteni és a rekordhalmazhoz adni. A rekordon végzett módosítások elvesznek:

- Ha az **Edit** vagy **AddNew** metódust használtuk, és azután az **Update** használata nélkül másik rekordra léptünk át.
- Ha az **Edit** vagy **AddNew** metódust használtuk, és azután az **Update** használata nélkül ismét egy **Edit** vagy **AddNew** metódust hívunk meg.
- Ha a **Bookmark** tulajdonságot másik rekordra állítjuk.
- Ha a *rekordhalmaz* azonosítóban adott nevű objektumot az **Update** használata nélkül zárjuk be.

Ha az **AddNew** metódust használjuk, és a Microsoft Jet adatbázis-szerkesztőnek egy 2K méretű lapot kell létrehozni az aktuális rekord megtartásához, az oldal zárolása

pesszimista, míg ha az új rekord illik a meglévő oldalhoz, optimista zárolású lesz.

Ha a eredményhalmaz halmaznak még nem az utolsó rekordján lennénk, az tartalmazhatja az alaptáblához adott rekordokat, ha azok az éppen az aktuális rekord utánra vannak pozícionálva. Ha a eredményhalmaz halmazon az **AddNew** metódust használva egy rekordot adunk a eredményhalmaz típusú **Recordset** objektumhoz, a rekord látható lesz a **Recordset** objektumban, és az alapjául szolgáló tábla részévé válik, ahol minden új **Recordset** objektum számára is láthatóvá válik.

Az új rekordnak a rekordhalmazon belüli helyzete a **Recordset** típusától függ:

- Egy eredményhalmaz típusú **Recordset** (vagy az elavult **Dynaset**) objektumban a rekordok a rekordhalmaz végéhez kapcsolódnak, függetlenül bármely, a rekordhalmaz megnyitásakor hatályos rendezési előírástól.
- Egy beállított **Index** tulajdonságú, tábla típusú **Recordset** (vagy az elavult **Table**) objektumban a rekordok visszaadása a rendezési előírások szerinti sorrendben történik.

Az **AddNew** végrehajtása előtt érvényes rekord továbbra is érvényes marad. Ha egy új rekordot akarunk aktuálissá tenni, a **Bookmark** tulajdonságot a **LastModified** tulajdonság értékével azonosított könyvjelzőhöz kell beállítani.

Egy cikluson belül bármely, egynél több rekordot szerkesztő DAO (Data Access Object) metódus használatakor (**AddNew**, **Delete**, **Edit** vagy **Update**) a Microsoft Jet 3.0-s verzió a teljes ciklust egy implicit tranzakcióként kezeli, vagyis minden egyes frissítést nem szükségszerűen írhatjuk azonnal az adatbázisba, ami hatást gyakorolhat a hibakezelésre, az oldalzárolásra és a változásoknak a többi felhasználó általi láthatóságára. A változások azonnali adatbázisba írásához egy explicit tranzakción belül minden egyes **Edit...Update** blokkot be kell zárni.

Append

Szintaxis: *objektumgyűjtemény*.**Append** *objektum*

Ez a **Containers**, **Databases**, **Documents**, **Fields**, **Groups**, **Indexes**, **Parameters**, **Properties**, **QueryDefs**, **Recordset**, **Relations**, **SelBookmarks**, **TableDefs** és **Users** objektumgyűjteményeken használható metódus egy új adathozzáférési objektumot ad egy objektumgyűjteményhez. A metódus szintaxisa a következő objektumazonosítókat tartalmazza:

- Az *objektumgyűjtemény* azonosító bármely, új objektumokat elfogadó halmaz lehet.
- Az *objektum* azonosítója egy, a hozzáadandó objektumot azonosító objektumtípus változója lehet, amelynek az *objektumgyűjtemény* argumentum elemeinek típusával kell azonosnak lennie.

Az **Append** metódus használata egy adatbázishoz egy új táblának egy adatbázishoz, egy mezőnek egy táblához és egy mezőnek egy indexhez adását foglalja magában. A csatolt objektum a **Delete** metódussal történő törlésig állandó objektummá válik, és lemezen tárolódik. Ha az *objektumgyűjtemény* egy **Workspaces** halmaz (amely csak a memóriában tárolódik), az objektum a **Close** metódussal történő eltávolításig aktív marad.

Egy új objektum hozzáadása azonnal bekövetkezik, azonban a **Refresh** metódus bármely más halmazon használható, amelyre az adatbázis szerkezetének változása hatással lehet.

Ha a hozzáadandó objektum nem teljes, vagy ha egy vagy több alárendelt objektum tulajdonságának beállítása helytelen, az **Append** metódus egy belső, lekezelhető hibajelzést indít el.

A következő táblázat az első oszlopban szereplő objektumnak a második oszlop objektumgyűjteményéhez **Append** metódussal történő hozzáadásának korlátait tartalmazza:

objektum:	objektumgyűjtemény:	Az **Append** használhatósága:
befogadó	**Documents**	Soha
Database	**Containers**	Soha
Database	**Recordsets**	Soha, helyette az **OpenRecordset** metódus használható
Database	**Properties**	Ha a **Database** állandó
Field	**Properties**	Ha a **Field** állandó
Index	**Fields**	Ha az **Index** új, még nem csatolt objektum
Index	**Properties**	Ha az **Index** állandó
QueryDef	**Fields**	Soha
QueryDef	**Parameters**	Soha
QueryDef	**Properties**	Ha a **QueryDef** állandó
Recordset	**Fields**	Soha
Relation	**Fields**	Soha
TableDef	**Fields**	Ha a **TableDef** **Updatable** tulajdonsága **True**
TableDef	**Indexes**	Ha a **TableDef** **Updatable** tulajdonsága **True**
TableDef	**Properties**	Ha a **TableDef** állandó
Workspace	**Databases**	Soha, helyette az **OpenDatabase** metódus használható

AppendChunk

Szintaxis: *rekordhalmaz*!*mező*.**AppendChunk** *forrás*

A **Field** objektumon értelmezett metódus egy karakteres kifejezésből adatokat ad a **Recordset** objektum **Fields** halmaza Memo vagy **OLE** objektumú eleméhez. A metódus

szintaxisa a következő objektumazonosítókat és argumentumot tartalmazza:

- A *rekordhalmaz* azonosító egy objektumtípusú változó, amely a **Recordset** objektumban foglalt **Fields** halmazra hivatkozik.
- A *mező* azonosító egy **Field** objektum, amelynek **Type** tulajdonsága **dbMemo** (Memo), **dbLongBinary** (OLE Object) vagy ezekkel ekvivalens érték.
- A *forrás* argumentum egy, a *mező* azonosítóval adott **Field** objektumhoz adandó adatot tartalmazó karakteres kifejezés vagy változó.

Az **AppendChunk** és **GetChunk** metódusok a **String** típusok 64K méretkorlátnál nagyobb méretű OLE Object mezőben lévő adatokhoz történő hozzáférésre használhatók. Ugyanezen metódusok az 1,2 GB méretkorlátú Memo mezőkkel történő munkák során a karakterlánc területének megőrzésére alkalmazhatók. Egyes műveletek — így például a másolás — ideiglenes karakterláncok használatával járnak. Ha a karakterlánc területe korlátozott, a teljes mező helyett csak Memo meződarabokkal dolgozhatunk.

A metódus használatakor az aktuális rekord hiánya hibát okoz.

Megjegyzendő, hogy a kezdő — az első **Edit** metódus utáni — **AppendChunk** metódus az adatot még akkor is egyetlenként helyezi el a mezőben, ha a rekord már tartalmaz adatot. Az egyetlen **Edit** munkameneten belül további **AppendChunk** hívások ezután már adathozzáfűzést hajtanak végre.

Arrange

Szintaxis: *objektum*.**Arrange** *elrendezés*

Az **MDIForm** objektumon érvényes **Arrange** metódussal az objektumon belüli ablakok vagy ikonok rendezhetők el. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *elrendezés* argumentum egy érték vagy változó, amely az ablakok vagy ikonok **MDIForm** objektumon belüli elrendezését írja elő.

Az *elrendezés* szimbolikus konstansa vagy értéke a következő táblázat szerinti lehet:

Ha a szimbolikus konstans:	Értéke:	Hatása:
vbCascade	0	Egymást átfedően rendezi el a nem ikonállapotú MDI-alablakokat
vbTileHorizontal	1	Vízszintesen egymás mellé rendezi a nem ikonállapotú MDI-alablakokat
vbTileVertical	2	Függőlegesen egymás alá rendezi a nem ikonállapotú MDI-alablakokat
vbArrangeIcons	3	Elrendezi az ikonállapotú MDI-alablakok ikonjait

Az **Arrange** metódus az alablakok és ikonok elrendezését az MDI-ablak ikonállapotában is elvégzi, de az eredmény természetesen csak az ablak teljes méretűvé tételekor lesz látható.

BeginTrans

Szintaxis 1: *munkaterület*.**BeginTrans**

Szintaxis 2: *adatbázis*.**BeginTrans**

Ez a **Workspace** objektumon használható metódus az objektum által meghatározott munkamenet során egy új tranzakciós folyamatot indít el. Noha a metódus a **Database** objektumon is használható, ez nem javasolt, mivel ezek a **Database** metódusok elavultak. A metódus szintaxisai a következő objektumazonosítót tartalmazzák:

- A *munkaterület* azonosító a **Workspace** objektumot megadó objektumtípusú változó.
- Az *adatbázis* azonosító a **Database** objektumot megadó objektumtípusú változó.

A metódust a **Workspace** vagy **Database** objektumokkal akkor használhatjuk, ha egyetlen munkamenetben változtatások sorozatot kívánunk elvégezni. A **Database** objektummal történő használata esetén a metódus a **Database** objektum **Workspace** objektumán alkalmazódik. A hasonló nevű tranzakciós utasítás használata nem javasolt.

Jellemzően a tranzakciókat két vagy több tábla rekordjainak frissítése során az adatok sértetlenségének megtartásához használjuk. A tranzakció megkezdéséhez a **BeginTrans** metódust az első rekord frissítése előtt kell használni.

Egy **Workspace** objektumon belül a tranzakciók globálisak a **Workspace** objektumra nézve, és nem csak egy adatbázisra vagy rekordhalmazra korlátozottak.

A tranzakciók öt szintje lehet egyszerre nyitott egy **Workspace** objektumban, többszörösen egymásba ágyazott **BeginTrans** és **CommitTrans** vagy **Rollback** kombinációk használata esetén. Külső ODBC SQL adatbázis nem teszi lehetővé egymásba ágyazott tranzakciók használatát.

A **CommitTrans** vagy **RollBack** metódusok használatát megelőzően egy **BeginTrans** metódus kell használni, különben hiba következik be.

A tranzakció megkezdése előtt szükséges annak ellenőrzése, hogy a használt adatbázis támogatja-e a tranzakciókat. Egynél több adatbázison alapuló **Recordset** objektum használatánál ez a **Recordset** objektum **Transaction** tulajdonságának vizsgálatával történik. Ha a rekordhalmaz teljesen Microsoft Jet adatbázis táblákon alapul, a **Transaction** tulajdonság **True** (-1) értékű, és a tranzakciók használhatók. A eredményhalmaz típusú, más adatbázisok által készített táblákon alapuló **Recordset** objektumok azonban nem támogatják a tranzakciókat, a **Transaction** tulajdonság tehát **False** (0) értékű lesz. Ilyen esetben a metódus használata nem okoz hibát, azonban hatástalan lesz.

Egy cikluson belül bármely, egynél több rekordot szerkesztő DAO (Data Access

Object) metódus használatakor (**AddNew**, **Delete**, **Edit** vagy **Update**) a Microsoft Jet 3.0-s verzió a teljes ciklust egy implicit tranzakcióként kezeli, vagyis minden egyes frissítést nem szükségszerűen írhatjuk azonnal az adatbázisba, ami hatást gyakorolhat a hibakezelésre, az oldalzárolásra és a változásoknak a többi felhasználó általi láthatóságára. A változások azonnali adatbázisba írásához egy explicit tranzakción belül minden egyes **Edit...Update** blokkot be kell zárni.

A tranzakciók az adatok sértetlenségének megtartásán kívül az alkalmazások jellemzőinek javítására is használhatók, mivel így az adatbázison lemezhozzáférést igénylő műveletek tranzakciós blokkokra tördelhetők, és ezzel a műveletek pufferelését biztosítva lényegesen csökkenthető a lemezhez fordulások száma.

Egy tranzakció megkezdésekor a Jet a műveleteit egy, a munkaállomás TEMP környezeti változójában megadott fájlba rögzíti. Ha a tranzakció naplózása során a TEMP meghajtóján rendelkezésre álló területet betelik, hiba (2004) következik be.

CancelUpdate

Szintaxis: *rekordhalmaz*.**CancelUpdate**

Ez a **Recordset** objektumon használható metódus megszakítja az objektum frissítését. A metódus szintaxisában a *rekordhalmaz* azonosítója egy objektumot kijelölő objektumkifejezés.

Az **Edit** vagy **AddNew** metódus hatására folyó, az **Update** metódussal még nem véglegesített frissítést a **CancelUpdate** metódus érvényteleníti, azaz hatása azonos az **Update** metódus használata nélkül egy másik rekordra átlépéssel, azonban itt az aktuális rekord nem változik és a különböző tulajdonságok — így például a **BOF** és **EOF** — sem frissítődnek.

CellText

Szintaxis: *objektum*. **CellText** *könyvjelző*

Ez a **Column** objektumon használható metódus egy **DBGrid** vezérlőelemcella tartalmát formázott szövegként adja vissza. A szintaxis a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *könyvjelző* argumentum egy, a **DBGrid** vezérlőelem egy sorát kijelölő karakteres kifejezés.

A metódus a kiválasztott vezérlőelem-cellában található adat karakteres, a **Column** objektum **NumberFormat** tulajdonságában meghatározott formázású alakját adja vissza. A visszaadott adat a **Text** tulajdonságból származik, vagyis a metódus hatása hasonló a **Text** tulajdonság kiolvasásához, azzal az eltéréssel, hogy itt kijelölhető a kiolvasandó sor

is: egy adott **DBGrid** vezérlőelem **Col** tulajdonságával a kívánt oszlop, a *könyvjelző* argumentummal pedig a kívánt rekord (sorra) állítható be, ezzel egy cellát kijelölve. A **CellValue** metódus használata nem hat az aktuális kiválasztásra.

CellValue

Szintaxis: *objektum.* **CellValue** *könyvjelző*

Ez a **Column** objektumon használható metódus egy **DBGrid** vezérlőelemcella formázatlan tartalmát adja vissza. A szintaxis a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *könyvjelző* argumentum egy, a **DBGrid** vezérlőelem egy sorát kijelölő karakteres kifejezés.

A visszaadott adat a **Value** tulajdonságból származik, vagyis a metódus hatása hasonló a **Value** tulajdonság kiolvasásához, azzal az eltéréssel, hogy itt kijelölhető a kiolvasandó sor is: egy adott **DBGrid** vezérlőelem **Col** tulajdonságával a kívánt oszlop, a *könyvjelző* argumentummal pedig a kívánt rekord (sorra) állítható be, ezzel egy cellát kijelölve. A **CellText** metódus használata nem hat az aktuális kiválasztásra.

Circle

Szintaxis: [*objektum*].**Circle** [**Step**] (*x*, *y*), *sugár* [, *szín*] [, *kezdet*] [, *vég*] [, *arány*]

Ez a **Form** és **Printer** objektumon, valamint **PictureBox** vezérlőelemen használható metódus az argumentumban szereplő értékeknek megfelelő kört, ellipszist vagy ívet rajzol. A szintaxis a következő objektumazonosítót és argumentumokat tartalmazza:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha nincs megadva, az *objektum* az éppen fókuszban lévő **Form** lesz.
- A feltételes **Step** kulcsszóval azt jelezhetjük, hogy a kör, ellipszis vagy ív középpontját az *objektum* **CurrentX** és **CurrentY** tulajdonságokkal adott aktuális koordinátáihoz viszonyítjuk.
- Az *x* és *y* **Single** típusú argumentumok a kör, ellipszis vagy ív középpontjának koordinátáit adják meg. Az *x* a vízszintes, az *y* pedig a függőleges koordinátát jelöli. A koordináták mérésére használt egységet az *objektum* **ScaleMode** tulajdonsága határozza meg. A rajzolás során a **CurrentX** és **CurrentY** tulajdonságok az *x* és *y* argumentumokkal adott középpontnak megfelelő értéket kapják.
- A feltételes, ugyancsak **Single** típusú *sugár* argumentum a kör, ellipszis vagy ív sugarát határozza meg. A *sugár* mérésére használt egységet most is az objektum **ScaleMode** tulajdonsága határozza meg.

- A feltételes, **Long** típusú *szín* argumentum a kör, ellipszis vagy ív RGB színét állítja be. A szín az **RGB( )** vagy a **QBColor( )** függvénnyel határozható meg. Hiányzó *szín* argumentum esetén a rajzszín a **ForeColor** tulajdonságának megfelelő lesz. A zárt alakzatok (kör vagy ellipszis) belsejének színét az *objektum* **FillColor** és **FillStyle** tulajdonságai határozzák meg.
- A feltételes, **Single** típusú *kezdet* és *vég* argumentum egy, a körből vagy ellipszisből kivágott ív radiánban adott kezdő- és végpontját jelölik ki. Alapértelmezetten a *kezdet* 0 (0 fok), a *vég* pedig 2*π (360 fok), azaz egy teljes kör. A *kezdet* és *vég* érvényességi tartománya -2*π és 2*π. A rajzolás a kezdőponttól a végpontig mindig az óra járásával ellentétes (pozitív) irányban történik. Ha a *kezdet* negatív, a **Circle** a kör- vagy ellipsziscikk kezdőpontjához, ha a *vég* negatív, a végpontjához rajzol sugarat, a szög értékét pedig mindkét esetben pozitívnak tekinti. Egy 0 foktól induló körcikk rajzolásához a körívet egy kis negatív értéktől kell indítani.
- A feltételes, **Single** típusú *arány* argumentumban az ellipszis rajzolásnál annak a körhöz viszonyított aránya adható meg. Alapértelmezett értéke 1, amely kör rajzolásával egyenértékű.

A kör, ellipszis vagy ív vonalvastagsága a **DrawWidth**, a háttér előtti megjelenés módja pedig a **DrawMode** és **DrawStyle** tulajdonságok értékének megfelelő lesz.

Az egyes feltételes argumentumok elhagyásakor az elválasztó vesszőt (az utolsó megadott argumentumot követőket kivéve) ki kell tenni.

Clear

Szintaxis: *objektum*.**Clear**

Ezzel a **Clipboard** és **Err** objektumon, valamint **ComboBox** és **ListBox** vezérlőelemen használható metódussal a vezérlőelem, illetve a rendszer vágólapjának tartalmát, az **Err** objektumon pedig az összes tulajdonság beállítását lehet törölni. Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Data** vezérlőelemhez kötött **ComboBox** vagy **ListBox** nem támogatja a **Clear** metódust.

A **Clear** metódussal egy hiba lekezelése után az **Err** objektum explicit módon törölhető, például az **On Error Resume Next** utasítással történő késleltetett hibakezelés esetén. (A **Clear** a Visual Basic előző verzióiban az `Err = 0` alakú utasításnak felelt meg.)

A Visual Basic a következő utasítások végrehajtásakor a **Clear** metódust automatikusan meghívja:

- Bármely alakú **Resume** utasítás
- **Exit Sub**, **Exit Function** vagy **Exit Property** utasítás

- Bármelyik alakú **On Error** utasítás

A más objektumokhoz történő hozzáférés során bekövetkező hibák kezeléséhez az **On Error GoTo** helyett az **On Error Resume Next** utasítás használata javasolt. Egy objektummal történő minden egyes művelet után az **Err** ellenőrzésével egyértelművé tehető, hogy a programkód éppen melyik objektumot használta a hiba bekövetkeztekor, és melyik (az **Err.Source** tulajdonságban található) objektum okozta a hibát és helyezte a hibaszámot az **Err.Number** tulajdonságba.

Clone

Szintaxis: **Set** *másolat* = *eredeti*.**Clone**( )

Ezzel a **Recordset** (továbbá az elavult **Dynaset**, **Snapshot** és **Table**) objektumon használható metódussal egy objektumról az eredeti objektumra hivatkozó másolat készíthető, anélkül, hogy az adatok az objektumban vagy az alapjául szolgáló struktúrában változnának. A szintaxisban a *másolat* argumentum a létrehozandó másolatot azonosító objektumtípusú változó, az *eredeti* argumentum pedig a másolandó objektumot azonosító objektumtípusú változó.

A **Clone** metódust a **Recordset** (továbbá az elavult **Dynaset**, **Snapshot** vagy **Table**) rekordok halmazán többszörös aktuális rekordot igénylő műveletek elvégzése esetén, a szükséges második rekordhalmaz létrehozása helyett a gyorsabb és hatékonyabb másolatkészítésre használhatjuk. Az így létrehozott minden egyes rekordhalmaz-másolatnak saját aktuális rekordja lehet, ha a másolatok használata előtt azokat a **Bookmark** tulajdonsággal, a eredményhalmaz vagy pillanatfelvétel típusú **Recordset**, (valamint az elavult **Dynaset** és **Snapshot**) objektumok esetén a **Move** metódusok egyikével, tábla típusú **Recordset** (vagy az elavult **Table**) objektumok esetén pedig a **Seek** metódussal beállítjuk.

A metódus a tábla típusú **Recordset** objektumon az **Index** tulajdonságot nem másolja, annak értékét manuálisan kell átmásolnunk.

A **Clone** használatával a könyvjelzők két vagy több rekordhalmaz között megosztottan is használhatók, mivel ezek könyvjelzői egymás között cserélhetők.

Egy **QueryDef** objektum futtatásának eredményeként kapott rekordhalmazról készített másolat a lekérdezést nem tudja ismét futtatni.

Ha a **Close** metódust akár az eredeti, akár a **Clone** metódussal létrehozott másolati objektumon használjuk, annak nem lesz hatása a másik objektumra.

A metódus nem használható az olyan pillanatfelvétel típusú **Recordset** objektummal, amelynek **dbForwardOnly** tulajdonsága beállított.

Close

Szintaxis: *objektum*.**Close**

A metódussal egy **OLE** befogadó vezérléselem bezárható és az objektumot szolgáltató alkalmazással fennálló kapcsolat megszakítható, illetve egy megnyitott **Database**, **QueryDef**, **Recordset**, **Workspace** (valamint az elavult **Dynaset**, **Snapshot** és **Table**) adathozzáférési objektum bezárható.

Az **OLE** befogadó vezérléselemen az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. A metódus csak beágyazott objektumokra vonatkozik, és egyenértékű az objektum bezárásával. A csatolt objektumokra nincs hatása.

A **Database**, **QueryDef**, **Recordset** vagy **Workspace** (valamint az elavult **Dynaset**, **Snapshot** vagy **Table**) objektumon az *objektum* azonosítója egy megnyitott objektum nevét adja meg. A megnyitott objektum bezárása eltávolítja azt abból a halmazból, amelyhez hozzá volt kapcsolva. Az alapértelmezett munkaterület bezárásának kísérlete figyelmen kívül marad.

A halmazból a **Database**, **Recordset** vagy **Workspace** objektumon kívüli objektum eltávolításához a **Delete** metódust kell használni.

Egy adatbázis bezárása előtt minden megnyitott rekordhalmaz objektumon az **Update** és a **Close** metódust kell használni. Rekordhalmazt vagy **Database** objektumokat deklaráló eljárásból kilépve az adatbázis bezáródik, minden nem mentett változás és folyamatban lévő adatszerkesztés elvész, továbbá minden folyamatban lévő tranzakció visszavonódik.

Ha olyankor kísérelünk meg bezárni egy **Database** objektumot, amíg bármely rekordhalmaz nyitott, vagy egy **Workspace** objektumot, amíg az ahhoz tartozó bármelyik **Database** objektum nyitott, a **Recordset** objektumok bezáródnak, és minden folyamatban lévő frissítés és szerkesztés visszavonódik.

Ha olyankor kísérelünk meg bezárni egy **Workspace** objektumot, amíg az ahhoz tartozó bármelyik **Database** objektum nyitott, a művelet az adott **Workspace** objektumhoz tartozó összes **Database** objektumot bezárja, és ezzel a bezáratlan **Recordset** objektumok bezáródását eredményezheti.

Ha a **Database** objektum az eljárás hatókörén kívül van definiálva, és az eljárásból annak bezárása nélkül lépünk ki, a **Database** az explicit bezárásig vagy az azt definiáló modul hatókörön kívül maradásáig nyitott marad.

A **Close** metódusnak az *objektum* argumentumban adott nevű, már bezárt rekordkészleten, **Database**, **Recordset** vagy **Workspace** objektumon történő használata kezelhető hibát okoz.

Ha a **Close** metódust akár az eredeti, akár a **Clone** metódussal létrehozott másolati objektumon használjuk, annak nem lesz hatása a másik objektumra.

A metódus nem használható az olyan pillanatfelvétel típusú **Recordset** objektummal, amelynek **dbForwardOnly** tulajdonsága beállított.

Cls

Szintaxis: [*objektum*].**Cls**

Ez a metódus a **Form** objektumról és **PictureBox** vezérlőelemről törli a futásidőben grafikus vagy nyomtató utasításokkal létrehozott grafikát és szöveget. A szintaxisban a feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha hiányzik, az *objektum* alapértelmezetten az éppen fókuszban lévő **Form** objektum lesz.

A **Form** objektumon a **Picture** tulajdonsággal létrehozott bitképes háttérrajzok és a programfejlesztési fázisban elhelyezett vezérlőelemek nem törlődnek. A **Cls** metódus ugyancsak érintetlenül hagyja az olyan, az **AutoRedraw** tulajdonság **True** értékénél a **Form** vagy **PictureBox** objektumba helyezett szöveget vagy grafikát, melyekre még a **Cls** végrehajtása előtt az **AutoRedraw** tulajdonságot **False** értékre állítjuk, így megőrizve azokat.

A **Cls** metódus végrehajtása után az *objektum* **CurrentX** és **CurrentY** tulajdonságai 0 értéket kapnak.

ColContaining

Szintaxis: *objektum*.**ColContaining** *koordináta*

Ez a metódus a **DBGrid** vezérlőelemen használható, és a vezérlőelem egy oszlopának **ColIndex** értékét adja vissza. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *koordináta* argumentum egy, a befogadó koordináta rendszerében egy vízszintes (X) koordinátát meghatározó numerikus kifejezés. A befogadó koordináta rendszerén kívüli *koordináta* megadása kezelhető hibát okoz.

A metódus által visszaadott szám az *objektum* vezérlőelem egyik oszlopindexének értéke, amely a **Columns** halmazon érvényes **LBound** és **UBound** tulajdonságok értékei között lehet. A metódus a vezérlőelem rejtett oszlopait is figyelembe veszi a befogadó oszlopindexének meghatározása során.

A metódust az egérrel és **DragDrop** vagy **DragOver** eseményekkel dolgozva annak vizsgálatára használhatjuk, hogy a felhasználó a **DBGrid** vezérlőelem oszlopában kifejezve hol kattintott vagy ejtett el egy másik vezérlőelemet.

CommitTrans

Szintaxis 1: *munkaterület*.**CommitTrans**

Szintaxis 2: *adatbázis*.**CommitTrans**

Ez a **Workspace** objektumon használható metódus az objektum által meghatározott

munkamenet során lezár egy tranzakciót és a változásokat menti. Noha a metódus a **Database** objektumon is használható, ez nem javasolt, mivel ezek a **Database** metódusok elavultak. A metódus szintaxisai a következő objektumazonosítókat tartalmazzák:

- A *munkaterület* azonosító a **Workspace** objektumot megadó objektumtípusú változó.
- Az *adatbázis* azonosító a **Database** objektumot megadó objektumtípusú változó.

A metódust a **Workspace** vagy **Database** objektumokkal akkor használhatjuk, ha az egyetlen munkamenetben végzett változtatások sorozatát az összes táblára véglegesíteni akarjuk. A **Database** objektummal történő használata esetén a metódus a **Database** objektum **Workspace** objektumán alkalmazódik. A hasonló nevű tranzakciós utasítás használata nem javasolt.

Jellemzően a tranzakciókat két vagy több tábla rekordjainak frissítése során az adatok sértetlenségének megtartásához használjuk. A tranzakció lezárása az utolsó rekord sikeres frissítése után használt **CommitTrans** metódussal történik.

Egy **Workspace** objektumon belül a tranzakciók globálisak a **Workspace** objektumra nézve, és nem csak egy adatbázisra vagy rekordhalmazra korlátozottak.

Ha az egynél több adatbázison vagy rekordkészleten hajtunk végre műveleteket, egy tranzakción belül a változásokat a **Rollback** metódussal lehet érvényteleníteni.

A **CommitTrans** használatát követően az adatbázison abban a munkamenetben végzett változtatások már nem vonhatók vissza, hacsak a tranzakció nem egy másik tranzakcióba van ágyazva, amelyben a **Rollback** metódussal a visszavonás még lehetséges. Egy **Workspace** objektumban, többszörösen egymásba ágyazott **BeginTrans** és **CommitTrans** vagy **Rollback** kombinációk használata esetén a tranzakciók öt szintje lehet egyszerre nyitva. Az egymásba ágyazott tranzakciók esetén egy szinttel csak a tranzakció lezárása és a változások mentése, illetve visszavonása után lehet visszalépni. Ha egyidejű, átlapolt és nem egymásba ágyazott hatókörű tranzakcióra van szükség, a versengő tranzakciókat tartalmazó járulékos **Workspace** objektumok hozhatók létre. A külső ODBC SQL adatbázisok nem teszi lehetővé egymásba ágyazott tranzakciók használatát.

A **Workspace** objektumnak a függőben lévő tranzakciók mentése vagy visszavonás nélküli lezárása esetén a tranzakciók hatása automatikusan visszavonódik.

A **CommitTrans** metódus használatát megelőzően egy **BeginTrans** metódus kell használni, különben hiba következik be.

Egy cikluson belül bármely, egynél több rekordot szerkesztő DAO (Data Access Object) metódus használatakor (**AddNew**, **Delete**, **Edit** vagy **Update**) a Microsoft Jet 3.0-s verzió a teljes ciklust egy implicit tranzakcióként kezeli, vagyis minden egyes frissítést nem szükségszerűen írhatjuk azonnal az adatbázisba, ami hatást gyakorolhat a hibakezelésre, az oldalzárolásra és a változásoknak a többi felhasználó általi láthatóságára. A változások azonnali adatbázisba írásához egy explicit tranzakción belül minden egyes **Edit...Update** blokkot be kell zárni.

A tranzakciók az adatok sértetlenségének megtartásán kívül az alkalmazások jellemzőinek javítására is használhatók, mivel így az adatbázison lemezhozzáférést igénylő műveletek tranzakciós blokkokra tördelhetők, és ezzel a műveletek pufferelését biztosítva lényegesen csökkenthető a lemezhez fordulások száma.

Egy tranzakció megkezdésekor a Jet a műveleteit egy, a munkaállomás TEMP környezeti változójában megadott fájlba rögzíti. Ha a tranzakció naplózása során a TEMP meghajtóján rendelkezésre álló területet betelik, hiba (2004) következik be. Ha ilyenkor a **CommitTrans** metódus van használatban, meghatározhatatlan számú művelet végrehajtása történik meg, a fennmaradók azonban elvesznek, és a műveletet újból kell indítani. A tranzakciós naplózás a **Rollback** metódus használatával szabadítható fel, és a tranzakción belüli összes művelet visszavonódik.

CompactDatabase

Szintaxis: **DBEngine.CompactDatabase** *régi_adatbázis, új_adatbázis* [, *hely* [, *lehetőségek*]]

Ez a **DBEngine** objektumon használható metódus egy bezárt adatbázist másol, tömörít, és lehetőséget ad a verziószám, a karakteres összehasonlítási mód, valamint a titkosítás beállítására. A metódus szintaxisa a következő argumentumokat tartalmazza:

- A *régi_adatbázis* argumentum egy karakteres kifejezés, amely egy létező, bezárt adatbázis — esetleg a teljes elérési utat is tartalmazó — neve. Ha a fájlnévnek kiterjesztése is van, az is meg kell adni, például "C:\MYDB.MDB". Amennyiben a hálózat támogatja, a hálózati elérési utat például a "\\MYSERVER\MYSHARE\MYDIR\MYDB.MDB" alakban adhatjuk meg.
- Az *új_adatbázis* argumentum egy karakteres kifejezés, amely a létrehozandó, tömörített adatbázis — teljes elérési utat is tartalmazó — neve. Amennyiben a hálózat támogatja, a *régi_adatbázis* argumentumhoz hasonlóan itt is megadhatunk hálózati elérési utat. Az *új_adatbázis* argumentumban nem lehet a *régi_adatbázis* argumentumban adott fájt megadni.
- A feltételes *hely* argumentumban adott, szimbolikus konstanst tartalmazó karakteres kifejezéssel a létrehozandó új adatbázis karakteres összehasonlítási módját írhatjuk elő. Ha az argumentum hiányzik, az új adatbázisra érvényes összehasonlítási szabály azonos lesz a régi adatbáziséval.
- A feltételes *lehetőségek* argumentumban adott, szimbolikus konstanst vagy konstansok kombinációját tartalmazó karakteres kifejezéssel a létrehozandó új adatbázis titkosítási módját és az adatforma verzióját írhatjuk elő.

A metódus *hely* argumentumában a karakteres összehasonlítás módját a következő konstansok egyikének a szöveg **CollatingOrder** tulajdonságába történő beállításával lehet előírni:

Ha a szimbolikus konstans:	Az összehasonlítás módja:
dbLangGeneral	Az angol, német, francia, portugál, olasz és modern spanyol nyelvnek megfelelően.
dbLangArabic	Az arab nyelvnek megfelelően.
dbLangCyrillic	Az orosz nyelvnek megfelelően.
dbLangCzech	A cseh nyelvnek megfelelően.
dbLangDutch	A holland nyelvnek megfelelően.
dbLangGreek	A görög nyelvnek megfelelően.
dbLangHebrew	A héber nyelvnek megfelelően.
dbLangHungarian	A magyar nyelvnek megfelelően.
dbLangIcelandic	Az izlandi nyelvnek megfelelően.
dbLangNordic	Az északi nyelveknek megfelelően. (Csak a Jet 1.0-s verzióban.)
dbLangNorwdan	A norvég és dán nyelvnek megfelelően.
dbLangPolish	A lengyel nyelvnek megfelelően.
dbLangSpanish	A hagyományos spanyol nyelvnek megfelelően.
dbLangSwedfin	A svéd és finn nyelvnek megfelelően.
dbLangTurkish	A török nyelvnek megfelelően.

A metódus *lehetőségek* argumentumában a töredezettség megszüntetésével történő tömörítés során a titkosítás vagy a titkosítás feloldása a következő konstansok egyikének használatával írható elő:

Ha a szimbolikus konstans:	A titkosítás módja:
dbEncrypt	Az adatbázis titkosítása a tömörítés során
dbDecrypt	Az adatbázis titkosításának feloldása a tömörítés során

Ha a titkosításra vonatkozó konstans hiányzik a metódus argumentumai közül, vagy mindkettő meg van adva, a titkosítási mód a régi adatbázisra beállítottnak felel meg.

A metódus *lehetőségek* argumentumában az újonnan létrehozott, tömörített adatbázis adatformájának verziószáma is előírható a következő szimbolikus konstansok közül egyet megadva:

Ha a szimbolikus konstans:	Az adatforma verziója:
dbVersion10	A tömörítéskor a Jet 1.0-s verzióját használó adatbázis létrehozása
dbVersion11	A tömörítéskor a Jet 1.1-s verzióját használó adatbázis létrehozása
dbVersion25	A tömörítéskor a Jet 2.5-s verzióját használó adatbázis létrehozása
dbVersion30	A tömörítéskor a Jet 3.0-s verzióját használó adatbázis létrehozása

Ha az adatformára vonatkozó konstans hiányzik, az adatforma a régi adatbázisra

beállítottnak felel meg. Az új adatbázisra csak a régi adatbázis adatformájának megfelelő vagy annál későbbi verziót használhatunk. A beállítás csak az új adatbázisra érvényes, és hatástalan a Microsoft Acces által létrehozott objektumokra, így űrlapokra és jelentésekre.

Egy adatbázisban az adatok módosításakor a fájl töredezetté válik, és ezzel a szükségesnél nagyobb lemezterületet foglal el. Az adatbázis rendszeres tömörítésével a fájl töredezettsége megszüntethető és mérete is csökkenthető. Az adatbázis másolása és tömörítése során a verziószám, továbbá az összehasonlítási és a titkosítási mód is megváltoztatható.

A **CompactDatabase** metódus nem teljesen alakítja át a Microsoft Access egyik verzióját a másikra, csak az adat formája konvertálódik, a definiált objektumok nem. A Microsoft Access régebbi verzióiból a jelenlegire történő konvertálás az automatikus indítás ablak File menüjének Convert Database parancsával lehetséges. Ha a metódussal a Microsoft Jet 1.1-s vagy 1.0-s verzióját a 2.5-s vagy 3.0-s verziójára konvertáljuk, ahhoz csak a kompatibilitási rétegképet tartalmazó alkalmazások férhetnek hozzá. A régebbi, a Jet 1.1-s vagy 1.0-s verziójával történő munkákhoz fejlesztett alkalmazások a 2.5-s vagy 3.0-s verzióval is használhatók, azonban az adaptálás nem automatikus.

A metódus a régi adatbázis minden adatát és biztonsági jogosultságát átmásolja az új adatbázisba. Az eljárás során a tömörített adatbázis adatai folyamatos szervezettségűek lesznek. A tömörítés megkezdése előtt a régi adatbázist be kell zárni. A többfelhasználós környezetben a többi felhasználó a tömörítés ideje alatt nem férhet hozzá az adatbázishoz. Ha a *régi_adatbázis* nincs bezárva, vagy kizárólagos használata nem lehetséges, hiba következik be.

Mivel a **CompactDatabase** egy adatbázis-másolatot hoz létre, az eredeti és a másolati példány számára is elegendő lemezterületnek kell rendelkezésre állnia, egyébként a töredezettség megszüntetésével történő tömörítési művelet sikertelen lesz. A régi és az új adatbázis nem lehet ugyanazon lemezen.

A sikeres tömörítés után törölni kell a régi adatbázist, majd az újat át kell nevezni az eredeti fájlnévre.

Copy

Szintaxis: *objektum*.**Copy**

Ez a metódus egy, az **OLE** befogadó vezérlőelemen belüli objektumot a rendszer vágólapjára másol. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Egy objektumnak a rendszer vágólapjára másolásakor az objektumhoz tartozó összes adat és csatolási információ is a vágólapra kerül. A másolt objektum csatolt vagy beágyazott is lehet.

A metódust egy menüben az Edit Copy parancs támogatására lehet használni.

CopyQueryDef

Szintaxis: *rekordhalmaz*.**CopyQueryDef**

Ez a metódus a **Recordset** objektumon használható, és az objektum létrehozásához használt **QueryDef** objektum egy másolatát adja vissza. A szintaxisban a *rekordhalmaz* azonosítója egy objektumot kijelölő objektumkifejezés.

Mivel a **CopyQueryDef** metódus a **Recordset** létrehozásához használt **QueryDef** másolatát hozza létre, ha az nem volt használva a **Recordset** objektumhoz, a metódus által visszaadott érték nulla lesz.

CreateDatabase

Szintaxis: **Set** *adatbázis* = [*munkaterület*].**CreateDatabase** (*adatbázisnév*, *hely* [, *lehetőségek*])

Ez a metódus a **Workspace** objektumon létrehoz egy új **Database** objektumot, lemezre menti, végül a megnyitott **Database** objektumot adja vissza. A metódus szintaxisa a következő objektumazonosítókat és argumentumokat tartalmazza:

- Az *adatbázis* azonosító a létrehozandó **Database** objektumot megadó objektumtípusú változó.
- A feltételes *munkaterület* azonosító egy létező, az adatbázist majd tartalmazó **Workspace** objektumot kijelölő, objektumtípusú változó. Ha nincs megadva, az alapértelmezett **Workspace** lesz érvényes.
- Az *adatbázisnév* argumentumban szereplő karakteres kifejezés a létrehozandó adatbázis — esetleg a teljes elérési utat is tartalmazó — neve, például "C:\MYDB.MDB". Ha a fájlnévhez nem adjuk meg a kiterjesztést, az automatikusan hozzákapcsolódik a fájlnévhez. (A metódussal csak .MDB adatbázisfájlt lehet létrehozni.) Amennyiben a hálózat támogatja, a hálózati elérési utat például a "\\MYSERVER\MYSHARE\MYDIR\MYDB.MDB" alakban adhatjuk meg.
- A *hely* argumentumban adott, szimbolikus konstanst tartalmazó karakteres kifejezéssel a létrehozandó új adatbázis karakteres összehasonlítási módját kell előírni. A hiányzó argumentum hibát okoz.
- A feltételes *lehetőségek* argumentumban adott, szimbolikus konstanst vagy konstansok kombinációját tartalmazó karakteres kifejezéssel a létrehozandó új adatbázis titkosítási módját és az adatforma verzióját írhatjuk elő.

A metódus *hely* argumentumában a karakteres összehasonlítás módját a következő konstansok egyikének a szöveg **CollatingOrder** tulajdonságába történő beállításával lehet előírni:

Ha a szimbolikus konstans:	Az összehasonlítás módja:
dbLangGeneral	Az angol, német, francia, portugál, olasz és modern spanyol nyelvnek megfelelően.
dbLangArabic	Az arab nyelvnek megfelelően.
dbLangCyrillic	Az orosz nyelvnek megfelelően.
dbLangCzech	A cseh nyelvnek megfelelően.
dbLangDutch	A holland nyelvnek megfelelően.
dbLangGreek	A görög nyelvnek megfelelően.
dbLangHebrew	A héber nyelvnek megfelelően.
dbLangHungarian	A magyar nyelvnek megfelelően.
dbLangIcelandic	Az izlandi nyelvnek megfelelően.
dbLangNordic	Az északi nyelveknek megfelelően. (Csak a Jet 1.0-s verzióban.)
dbLangNorwdan	A norvég és dán nyelvnek megfelelően.
dbLangPolish	A lengyel nyelvnek megfelelően.
dbLangSpanish	A hagyományos spanyol nyelvnek megfelelően.
dbLangSwedfin	A svéd és finn nyelvnek megfelelően.
dbLangTurkish	A török nyelvnek megfelelően.

A metódus *lehetőségek* argumentumában a töredezettség megszüntetésével történő tömörítés során a titkosítás a következő konstans használatával írható elő:

Ha a szimbolikus konstans:	A titkosítás módja:
dbEncrypt	Titkosított adatbázis létrehozása

Ha a titkosításra vonatkozó konstans hiányzik a metódus argumentumai közül, az adatbázis nem titkosított lesz.

A metódus *lehetőségek* argumentumában az újonnan létrehozott, tömörített adatbázis adatformájának verziószáma is előírható a következő szimbolikus konstansok közül egyet megadva:

Ha a szimbolikus konstans:	Az adatforma verziója:
dbVersion10	A tömörítéskor a Jet 1.0-s verzióját használó adatbázis létrehozása
dbVersion11	A tömörítéskor a Jet 1.1-s verzióját használó adatbázis létrehozása
dbVersion25	A tömörítéskor a Jet 2.5-s verzióját használó adatbázis létrehozása
dbVersion30	A tömörítéskor a Jet 3.0-s verzióját használó adatbázis létrehozása

Ha a verziószámot előíró konstans hiányzik a metódus argumentumai közül, a használt környezettől függően a Microsoft Jet 2.5-s verziót (16-bites környezet) vagy a Jet 3.0-s verziót (32-bites környezet) használó adatbázis jön létre.

A **CreateDatabase** metódussal egy új adatbázissablon hozható létre, amelyet a metódus megnyit, és a **Database** objektumát adja vissza számunkra, amelynek szerkezetét és tartalmát további adathozzáférési objektumok hozzáadásával nekünk kell teljessé tenni.

Ha egy létező adatbázisról akarunk részleges vagy teljes másolatot készíteni, erre inkább a **CompactDatabase** metódust használhatjuk.

A Visual Basic előző verzióiban az adatbázisok létrehozására a **CreateDatabase** utasítást lehetett használni. Noha kompatibilitási okokból ez az utasítás a Visual Basic 4.0-s verziójában is megvan, azonban helyette az azonos nevű metódust használata szükséges.

CreateDynaset

Szintaxis 1: **Set** *dinamikus_objektum* = *adatbázis*.**CreateDynaset** (*forrás*, [, *lehetőségek*])

Szintaxis 2: **Set** *dinamikus_objektum* = [*rekordhalmaz* | *lekérdezésdefiníció*].**CreateDatabase** ([*lehetőségek*])

Ez a **Database**, **Dynaset**, **QueryDef**, **Snapshot** és **Table** objektumokon, valamint **QueryDef** objektumgyűjteményeken használható metódus egy új **Dynaset** objektumot hoz létre az adott **Dynaset**, **Table** vagy **QueryDef** objektumból, illetve SQL utasításból. A metódus szintaxisai a következő objektumazonosítókat és argumentumokat tartalmazzák:

- A *dinamikus_objektum* azonosítója egy **Dynaset** objektumváltozó.
- Az *adatbázis* azonosító egy nyitott adatbázist megadó objektumváltozó.
- A *rekordhalmaz* azonosító egy létező **Dynaset** vagy **Table** objektumot megadó objektumváltozó.
- A *lekérdezésdefiníció* azonosító egy létező **QueryDef** objektumot megadó objektumváltozó.
- A *forrás* argumentum egy létező rekordhalmaz, **QueryDef** objektum vagy SQL utasítás nevét tartalmazó karakteres kifejezés.
- A feltételes *lehetőségek* argumentum egy numerikus (vagy szimbolikus konstansokat tartalmazó) kifejezés, amely egy vagy több beállítást jelölhet ki.

A metódus *lehetőségek* argumentumban adott állandók vagy értékek hatása a következő táblázat szerinti lehet:

Ha a szimbolikus konstans:	Értéke:	Hatása:
dbDenyWrite	1	Többfelhasználós vagy többfeladatos környezetben mindaddig mások a **Dynaset** objektumunk semelyik táblájának semelyik rekordját sem módosíthatják, ameddig azt nyitva tartjuk. Csak adminisztratív célból használható.
dbReadOnly	4	A **Dynaset** objektum rekordjai nem módosíthatók. Használatával néhány művelet sebessége megnövelhető.
dbAppendOnly	8	A **Dynaset** objektumhoz csak új rekordokat adhatunk, olvasás és frissítés nem lehetséges.
dbInconsistent	16	A frissítés még akkor is csak a **Dynaset** objektum összes rekordján lehetséges, ha azok a **Dynaset** más soraira is hatással lehetnek.
dbConsistent	32	A frissítés csak a **Dynaset** objektum más soraira nem ható rekordjain lehetséges. Alapértelmezett.
dbSQLPassThrough	64	SQL utasítással létrehozott **Dynaset** objektumon ennél az értéknél az SQL utasítás feldolgozása egy ODBC adatbázisnak adódik át, egy SQL Server adatbázis tábláin a **Dynaset** objektum létrehozásakor. A **Dynaset** nem frissíthető. A **dbSQLPassThrough** beállítású **CreateDynaset** metódusnál hatékonyabb a eredményhalmaz típusú **Recordset** vagy **Snapshot** objektum használata. Csak távoli ODBC adatbázisok esetén hatásos.

Ha a **dbInconsistent** és **dbConsistent** állandók közül mindkettő vagy egyik sem szerepel a *lehetőségek* argumentumban, az eredmény az alapértelmezett **dbConsistent** lesz.

A Visual Basic 3.0-s verziójában lehetséges volt a **CreateDynaset** metódus rekordokat nem visszaadó SWLPassThrough lekérdezés végrehajtására történő használata, azonban ez a Visual Basic 4.0-s verzióban már nem lehetséges.

A **CreateDynaset** metódust **Dynaset** vagy **Snapshot** objektumon használva egy új, a **Dynaset** vagy **Snapshot** objektum **Filter** vagy **Sort** tulajdonságával szűrt rekordokat tartalmazó **Dynaset** objektumot kapunk vissza. A **CreateDynaset** hatástalan az első **Dynaset** vagy **Snapshot** tartalmára. Ezen tulajdonságok nem javasoltak a nagy rekordhalmazokhoz, egy SQL lekérdezés létrehozása hatékonyabb egy módosított **WHERE** vagy **ORDER BY** záradékkal.

A **Dynaset** objektumok használhatók az SQL utasításátadó műveletekkel, azonban a lekérdezésből csak egy eredményhalmazt kapunk vissza, és minden üzenet törlődik Egynél több eredmény- és üzenethalmazt átadó lekérdezés futtatása a **QueryDef** objektum használatával lehetséges.

Egy **QueryDef** objektumból egy paraméterekkel rendelkező **Dynaset** objektum létrehozásához meg kell nyitni a **QueryDef** objektumot, a paramétereket a *lekérdezésdefiníció!paraméter = paraméterérték* alakú kifejezéssel be kell állítani, majd a **CreateDynaset** metódust kell használni.

Ha az adatokat csak olvasni, de nem módosítani akarjuk, a **Snapshot** objektum gyorsabb, mint a csak olvasható **Dynaset** objektum. A csak olvasható **Dynaset** csak a mások által végzett változtatásokat tükrözi, amíg azt mi használjuk, de a **Snapshot** nem.

Ha a programunk számára két különböző **Dynaset** objektumra van szükség, hatékonyabb csak egy **Dynaset** objektumot létrehozni, majd a **Clone** metódussal egy második hivatkozást készíteni.

CreateEmbed

Szintaxis: *objektum*.**CreateEmbed** *forrásdokumentáció* [, *osztály*]

Ez a metódus **OLE** befogadó vezérlőelemen használható, és egy beágyazott objektumot hoz létre. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *forrásdokumentáció* argumentum a beágyazott objektum számára sablonként használható dokumentum kötelezően megadandó fájlneve. Ha nem akarunk forrásdokumentumot megadni, üres ("") karakterláncot kell használni.
- A feltételes *osztály* argumentum a beágyazott objektum osztályának neve. Ha a *forrásdokumentáció* számára fajlnevet adtunk meg, az argumentum tartalma közömbös.

A **CreateEmbed** metódus használata előtt az **OLETypeAllowed** tulajdonságot az *Embedded* (1) vagy az *Either* (2) értékre kell beállítani, de a **Class** és **SourceDoc** tulajdonságot nem szükséges megadni. A rendszerben érvényes osztálynevek listájának megtekintéséhez a Properties ablakban a Class tulajdonságot kell kiválasztani, majd a Properties nyomógombra kell kattintani.

Egy új objektum létrehozásakor az osztálynévvel kapcsolatos alkalmazásnak (például EXCEL.EXE) megfelelő módon — az alkalmazás telepítő programjával — regisztráltnak kell lennie az operációs rendszerben.

CreateField

Szintaxis: **Set** *változó* = *rekordhalmaz*.**CreateField** ([*név* [, *típus* [, *méret*]]])

Ez az **Index**, **Relation** és **TableDef** objektumokon érvényes metódus egy új **Field** objektumot hoz létre. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A *változó* argumentum egy **Field** objektumtípusként deklarált változó.
- A *rekordhalmaz* azonosítója az új **Field** objektum létrehozásához használni kívánt **Index**, **Relation** vagy **TableDef** objektum változóneve.
- Az új **Field** objektum egyedi nevét a feltételes *név* argumentum **String** típusú változójaként kell megadni. Ha a *név* olyan objektumra hivatkozik, amely már egy halmaz tagja, az **Append** metódus használata kezelhető hibát okoz. Az érvényes **Field** nevekről a **Name** tulajdonságánál található bővebb információ.
- A feltételes *típus* argumentum egy **Integer** típusú konstans, amely az új **Field** objektum adattípusát határozza meg. Az argumentum értéke a nem **TableDef** objektumban lévő **Field** objektumok (azaz **Index** és **Relation** objektumok) esetén közömbös. Az érvényes adattípusok a **Type** tulajdonságnál találhatók.
- A feltételes *méret* argumentum az új, szöveget tartalmazó **Field** objektum byteban mért hosszát megadó egész szám. A *méret* érvényes értékei a **Size** tulajdonságnál találhatók. Az argumentum értéke a nem **TableDef** objektumban lévő **Field** objektumok (azaz **Index** és **Relation** objektumok), továbbá numerikus és rögzített szélességű mezők esetén közömbös.

A **CreateField** metódusban egy vagy több feltételes argumentum elhagyásakor megfelelő beállítási utasításokkal az adott tulajdonság beállítható vagy módosítható. Ezt célszerű az új objektum objektumgyűjteményhez adása előtt megtenni, mivel a hozzáadás után már nem mindegyik tulajdonság értéke módosítható. Erről további részletek az egyes tulajdonságok ismertetésénél találhatók.

Egy halmazból **Field** objektumot általában a **Delete** metódussal távolíthatunk el, de ez egy **TableDef** objektum **Fields** halmazából a mezőre hivatkozó index létrehozását követően már nem lehetséges.

CreateGroup

Szintaxis: **Set** *változó* = *objektum*.**CreateGroup** ([*név* [, *pid*]])

Ez a **User** és **Workspace** objektumokon érvényes metódus egy új **Group** objektumot hoz létre. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A *változó* argumentum egy **Group** objektumtípusként deklarált változó.
- Az *objektum* azonosítója az új **Group** objektum létrehozásához használni kívánt **User** vagy **Workspace** objektum változóneve.
- Az új **Group** objektum egyedi nevét a feltételes *név* argumentum **String** típusú változójaként kell megadni. Ha a *név* olyan objektumra hivatkozik, amely már egy halmaz tagja, az **Append** metódus használata kezelhető hibát okoz. Az érvényes **Group** nevekről a **Name** tulajdonságánál található bővebb információ.
- A feltételes *pid* argumentum egy 4—20 alfanumerikus karakterből álló **String**

típusú, egy csoportszámla személyi azonosítókódjának (PID) azonosítóját tartalmazó változó. Az érvényes azonosítókódra vonatkozó további információk a **PID** tulajdonságnál találhatók.

A **CreateGroup** metódusban egy vagy több feltételes argumentum elhagyásakor megfelelő beállítási utasításokkal az adott tulajdonság beállítható vagy módosítható. Ezt célszerű az új objektum objektumgyűjteményhez adása előtt megtenni, mivel a hozzáadás után már nem mindegyik tulajdonság értéke módosítható. Erről további részletek az egyes tulajdonságok ismertetésénél találhatók.

Egy halmazból **Group** objektumot a **Delete** metódussal távolíthatunk el.

CreateIndex

Szintaxis: **Set** *változó* = *tábladefiníció*.**CreateIndex** ([*név*])

Ez a **TableDef** objektumon érvényes metódus egy új **Index** objektumot hoz létre. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- A *változó* argumentum egy **Index** objektumtípusként deklarált változó.
- A *tábladefiníció* argumentum az új **Index** objektum létrehozásához használni kívánt **TableDef** objektum változóneve.
- Az új **Index** objektum egyedi nevét a feltételes *név* argumentum **String** típusú változójaként kell megadni. Ha a *név* olyan objektumra hivatkozik, amely már egy halmaz tagja, az **Append** metódus használata kezelhető hibát okoz. Az érvényes **Index** nevekről a **Name** tulajdonságánál található bővebb információ.

A **CreateIndex** metódusban a feltételes *név* argumentum elhagyásakor megfelelő beállítási utasítással a **Name** tulajdonság beállítható vagy módosítható. Ezt célszerű az új objektum objektumgyűjteményhez adása előtt megtenni, mivel a hozzáadás után már csak attól az objektumtípustól függően módosítható, amelyben az **Indexes** halmaz rejlik. Erről további részletek a **Name** tulajdonság ismertetésénél találhatók.

Egy halmazból **Index** objektumot a **Delete** metódussal távolíthatunk el.

CreateLink

Szintaxis: *objektum*.**CreateLink** *forrásdokumentáció* [, *forráselem*]

Ez a metódus **OLE** befogadó vezérlőelemen használható, és egy fájl tartalmából csatolt objektumot hoz létre. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *forrásdokumentáció* argumentum az objektum létrehozásához használt fájlt határozza meg.

- A feltételes *forráselem* argumentum az a fájlon belüli adat, amelyhez a csatolt objektum csatolva van.

A **CreateLink** metódus használata előtt az **OLETypeAllowed** tulajdonságot a *Linked* (0) vagy az *Either* (2) értékre kell beállítani. Ha a metódus argumentumaihoz értékeket adunk meg, ezek felülírják a **Class** és **SourceDoc** tulajdonságok értékeit. A metódus meghívásakor a tulajdonságok az argumentumok értékeinek megfelelően frissítődnek.

Amikor ezzel a metódussal hozunk létre egy objektumot, az **OLE** befogadó vezérlőelem a **SourceDoc** tulajdonság által adott fájl hasonmását jeleníti meg. Az objektum mentésekor csak a csatolás hivatkozása mentődik, mivel az **OLE** befogadó vezérlőelem csak az adat metafájl hasonmását tartalmazza, az aktuális forrásadatokat nem.

Egy új objektum létrehozásakor az osztálynévvel kapcsolatos alkalmazásnak (például EXCEL.EXE) megfelelő módon — az alkalmazás telepítő programjával — regisztráltnak kell lennie az operációs rendszerben.

CreateProperty

Szintaxis: **Set** *változó* = *objektum*.**CreateProperty** ([*név* [, *típus* [, *érték* [, *DDL*]]]])

Ez a **Database**, **Field**, **Index**, **QueryDef** és **TableDef** objektumokon érvényes metódus egy új, felhasználó által definiált **Property** objektumot hoz létre. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A *változó* argumentum egy **Property** objektumtípusként deklarált változó.
- Az *objektum* azonosítója az új **Property** objektum létrehozásához használni kívánt **Database**, **Field**, **Index**, **QueryDef** vagy **TableDef** objektum változóneve.
- Az új **Property** objektum egyedi nevét a feltételes *név* argumentum **String** típusú változójaként kell megadni. Ha a *név* olyan objektumra hivatkozik, amely már egy halmaz tagja, az **Append** metódus használata kezelhető hibát okoz. Az érvényes **Property** nevekről a **Name** tulajdonságánál található bővebb információ.
- A feltételes *típus* argumentum egy **Integer** típusú konstans, amely az új **Property** objektum adattípusát határozza meg. Az érvényes adattípusok a **Type** tulajdonságnál találhatók.
- A feltételes *érték* argumentum a tulajdonság kezdeti értékét megadó **Variant** változó. az új, szöveget tartalmazó **Property** objektum byte-ban mért hosszát megadó egész szám. Az *érték* érvényes értékei a **Value** tulajdonságnál találhatók.
- A feltételes *DDL* argumentum egy **Boolean** típusú érték, amely jelzi, hogy a **Property** DDL objektum-e. Alapértelmezett értéke **False**. Ha az *DDL* értéke **True**, a **Property** objektum csak akkor módosítható vagy törölhető, ha a felhasználó **dbSecWriteDef** joggal rendelkezik.

Felhasználó által definiált **Property** objektum csak egy állandó (már lemezen tárolt) objektum **Properties** halmazán hozható létre.

A **CreateProperty** metódusban egy vagy több feltételes argumentum elhagyásakor megfelelő beállítási utasításokkal az adott tulajdonság beállítható vagy módosítható. Ezt célszerű az új objektum objektumgyűjteményhez adása előtt megtenni, mivel a hozzáadás után már nem mindegyik tulajdonság értéke módosítható. Erről további részletek az egyes tulajdonságok ismertetésénél találhatók.

Egy halmazból **Property** objektumot általában a **Delete** metódussal távolíthatunk el, de a belső, beépített tulajdonságok nem törölhetők.

A **Property** objektum DLL objektumról nem DLL objektumra történő változtatása csak az objektum törlésével és újbóli létrehozásával lehetséges.

CreateQueryDef

Szintaxis: **Set** *lekérdezésdefiníció* = *adatbázis*.**CreateQueryDef** ([*név*] [, *sqlszöveg*])

Ez a **Database** objektumon érvényes metódussal egy adott adatbázisban új **QueryDef** objektumot hozhatunk létre. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A *lekérdezésdefiníció* argumentum a létrehozandó **QueryDef** objektumot képviselő objektumtípusú változó.
- Az *adatbázis* azonosító az új **QueryDef** objektumot tartalmazó **Database** objektumot képviselő objektumtípusú változó.
- Az új **QueryDef** objektum nevét a feltételes *név* argumentum **String** típusú változójaként kell megadni. Ha az objektum már a **QueryDefs** halmaz egy tagja lenne, kezelhető hiba következik be.
- A feltételes *sqlszöveg* argumentum a **QueryDef** objektumot definiáló **String** típusú kifejezés (SQL utasítás). Elhagyása esetén a **QueryDef** az objektumgyűjteményhez adás előtt vagy után is definiálható az **SQL** tulajdonságának beállításával.

Ha az argumentumok közül a *név* megadott, az *sqlszöveg* pedig hiányzik, illetve ha mindkettő megadott, a **QueryDef** a létrehozáskor vagy módosításkor automatikusan mentődik.

A legutolsóként létrehozott **QueryDef** objektum **Name** tulajdonságába vagy a **CreateQueryDef** metódus végrehajtásakor a *név* argumentumba üres ("") karakterláncot beállítva ideiglenes **QueryDef** objektum hozható létre. Ez nem adható egyetlen objektumgyűjteményhez sem, mivel az üres karakterlánc nem lehet az állandó **QueryDef** objektumok érvényes neve, de a **Name** és **SQL** tulajdonságainak beállítását követően már az adatbázisban tárolt **QueryDefs** halmazhoz adható. A *név* argumentumban nem üres karakterlánccal létrehozott **QueryDef** objektumok automatikusan adódnak a **QueryDefs**

halmazhoz.

Egy **QueryDefs** halmazból **QueryDef** objektumot a **Delete** metódussal távolíthatunk el.

Egy SQL utasítás **QueryDef** objektumban történő végrehajtásához az **Execute** metódust kell használnunk.

Az SQL utasításokat átadó lekérdezések ODBC adatbázisokkal történő végrehajtására a **QueryDef** objektum használata javasolt. További információk a **QueryDef** objektum ismertetésénél találhatók.

CreateRelation

Szintaxis: **Set** *változó* = *adatbázis*.**CreateRelation** ([*név* [, *tábla* [, *idegen_tábla* [, *attribútumok*]]]])

Ez a **Database** objektumon érvényes metódus egy új **Relation** objektumot hoz létre. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A *változó* argumentum egy **Relation** objektumtípusként deklarált változó.
- Az *adatbázis* azonosító az új **Relation** objektum létrehozásához használt **Database** objektum változóneve.
- Az új **Relation** objektum egyedi nevét a feltételes *név* argumentum **String** típusú változójaként kell megadni. Ha a *név* olyan objektumra hivatkozik, amely már egy halmaz tagja, vagy ha az alárendelt **Fields** halmazban adott **Field** objektumnév érvénytelen, az **Append** metódus használata kezelhető hibát okoz. Az érvényes **Relation** nevekről a **Name** tulajdonságánál található bővebb információ.
- A feltételes *tábla* argumentum a relációban az elsődleges táblát megnevező **String** típusú változó. A **Relation** objektum hozzáadásakor még nem létező tábla kezelhető hibát okoz.
- A feltételes *idegen_tábla* argumentum a reláció idegen táblarészt megadó **String** típusú változó. A **Relation** objektum hozzáadásakor még nem létező tábla kezelhető hibát okoz.
- A feltételes *attribútumok* argumentum egy **Long** típusú szám, amely a viszonyra vonatkozó információkat hordozza. Az argumentumról az **Attributes** tulajdonságánál található bővebb információ.

A **Relation** objektum a Microsoft Jet adatbázis-szerkesztő számára adja tovább két **TableDef** vagy **QueryDef** objektum mezői között fennálló viszonyra vonatkozó információkat. A Jet az **Attributes** tulajdonságon keresztül biztosít hivatkozási integritást.

A **CreateRelation** metódusban egy vagy több feltételes argumentum elhagyásakor megfelelő beállítási utasításokkal az adott tulajdonság beállítható vagy módosítható. Ezt célszerű az új objektum objektumgyűjteményhez adása előtt megtenni, mivel a hozzáadás után már egyik tulajdonságának értéke sem módosítható. Erről további részletek az

egyes tulajdonságok ismertetésénél találhatók.

Az **Append** metódus **Relation** objektumon történő használata előtt hozzá kell adni a megfelelő **Field** objektumokat az elsődleges és idegen kulcs viszonylattábláinak definiálására.

Egy halmazból **Relation** objektumot a **Delete** metódussal távolíthatunk el.

CreateSnapshot

Szintaxis:

Ez a **Database**, **QueryDef**, (valamint az elavult **Dynaset**, **Snapshot** és **Table**) objektumokon, valamint **QueryDef** objektumgyűjteményeken használható, új **Snapshot** objektumot létrehozó metódus a Visual Basic 4.0-s verziójában csak kompatibilitási okokból maradt meg, helyette a pillanatfelvétel típusú **Recordset** objektumot létrehozó **OpenRecordset** metódus használata javasolt. Megjegyzendő, hogy a **CreateSnapshot** és az **OpenRecordset** metódussal létrehozható **Snapshot** objektumok nem teljesen azonosak, az **OpenRecordset** metódussal létrehozottak kiegészítő funkcionális tulajdonságokkal rendelkeznek.

CreateTableDef

Szintaxis: **Set** *változó* = *adatbázis*.**CreateTableDef** ([*név* [, *attribútumok* [, *forrás* [, *kapcsolat*]]]])

Ez a **Database** objektumon érvényes metódus egy új **TableDef** objektumot hoz létre. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A *változó* argumentum egy **TableDef** objektumtípusként deklarált változó.
- Az *adatbázis* azonosító az új **TableDef** objektum létrehozásához használt **Database** objektum változóneve.
- Az új **TableDef** objektum egyedi nevét a feltételes *név* argumentum **String** típusú változójaként kell megadni. Ha a *név* olyan objektumra hivatkozik, amely már egy halmaz tagja, vagy ha a **TableDef** vagy **Field** objektumban érvénytelen értéket adtunk meg, az **Append** metódus használata kezelhető hibát okoz. Az érvényes **TableDef** nevekről a **Name** tulajdonságánál található bővebb információ.
- A feltételes *attribútumok* argumentum egy **Long** típusú szám, amely az új **TableDef** objektum egy vagy több jellemzőjét jelöli. Az argumentumról az **Attributes** tulajdonságánál található bővebb információ.
- A feltételes *forrás* argumentum az eredeti adatforrást tartalmazó adatbázis nevét megadó **String** típusú változó. A *forrás* karakterlánc az új **TableDef** objektum **SourceTableName** tulajdonságává válik.
- A feltételes *kapcsolat* argumentum egy nyitott adatbázis forrásáról, átmenő le-

kérdezésben használt adatbázisról vagy egy csatolt táblára vonatkozó információkat tartalmazó **String** típusú változó. Az argumentum érvényes értékeiről a **Connect** tulajdonságánál található bővebb információ.

A **CreateTableDef** metódusban egy vagy több feltételes argumentum elhagyásakor megfelelő utasításokkal az adott tulajdonság beállítható vagy módosítható. Ezt célszerű még az új objektum objektumgyűjteményhez adása előtt megtenni, mivel a hozzáadás után már nem mindegyik tulajdonság értéke módosítható. Erről további részletek az egyes tulajdonságok ismertetésénél találhatók.

Egy halmazból **TableDef** objektumot a **Delete** metódussal távolíthatunk el.

CreateUser

Szintaxis: **Set** *változó* = *objektum*.**CreateGroup** ([*név* [, *pid* [, *jelszó*]]])

Ez a **Group** és **Workspace** objektumokon érvényes metódus egy új **User** objektumot hoz létre. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A *változó* argumentum egy **User** objektumtípusként deklarált változó.
- Az *objektum* azonosítója az új **User** objektum létrehozásához használni kívánt **Group** vagy **Workspace** objektum változóneve.
- Az új **Group** objektum egyedi nevét a feltételes *név* argumentum **String** típusú változójaként kell megadni. Ha a *név* olyan objektumra hivatkozik, amely már egy halmaz tagja, az **Append** metódus használata kezelhető hibát okoz. Az érvényes **User** nevekről a **Name** tulajdonságánál található bővebb információ.
- A feltételes *pid* argumentum egy 4—20 alfanumerikus karakterből álló **String** típusú, egy csoportszámla személyi azonosítókódjának (PID) azonosítóját tartalmazó változó. Az érvényes azonosítókódra vonatkozó további információk a **PID** tulajdonságnál találhatók.
- A feltételes *jelszó* argumentum egy legfeljebb 14 alfanumerikus karakterből álló **String** típusú, a felhasználói jelszót tartalmazó változó. A jelszóban a 0 (nulla) kivételével bármely ASCII karaktert használható.

A **CreateUser** metódusban egy vagy több feltételes argumentum elhagyásakor megfelelő utasításokkal az adott tulajdonság beállítható vagy módosítható. Ezt célszerű még az új objektum objektumgyűjteményhez adása előtt megtenni, mivel a hozzáadás után már nem mindegyik tulajdonság értéke módosítható. Erről további részletek az egyes tulajdonságok ismertetésénél találhatók.

Egy halmazból **User** objektumot a **Delete** metódussal távolíthatunk el.

CreateWorkspace

Szintaxis: **Set** *változó* = **CreateWorkspace** (*név*, *felhasználó*, *jelszó*)

Ez a **DBEngine** objektumon érvényes metódus egy új **Workspace** objektumot hoz létre. A metódus szintaxisa a következő argumentumokat tartalmazza:

- A *változó* argumentum egy **Workspace** objektumtípusként deklarált változó.
- Az új **Workspace** objektum egyedi nevét a *név* argumentum **String** típusú változójaként kell megadni. Ha a *név* olyan objektumra hivatkozik, amely már a **Workspaces** objektumgyűjtemény tagja, az **Append** metódus használata kezelhető hibát okoz. Az érvényes **Workspace** nevekről a **Name** tulajdonságánál található bővebb információ.
- A *felhasználó* argumentum az új **Workspace** objektum felhasználóját azonosító **String** típusú változó. További információk a **UserName** tulajdonságnál találhatók.
- A *jelszó* argumentum az új **Workspace** objektumhoz egy legfeljebb 14 alfanumerikus karakterből álló **String** típusú jelszót tartalmazó változó. A jelszóban a 0 (nulla) kivételével bármely ASCII karaktert használható.

Más, adathozzáférési objektum létrehozására használt metódusoktól eltérően a **CreateWorkspace** metódusban az összes argumentum megadása kötelező, továbbá a létrejövő objektum nem állandó és nem menthető. Miután a **Workspace** objektumot létrehoztunk, annak semelyik tulajdonsága sem módosítható.

A **Workspace** objektumot használata előtt csak akkor kell hozzáadni egy objektumgyűjteményhez, ha arra a **Workspaces** halmazon keresztül kívánunk hivatkozni.

A **CreateWorkspace** metódussal történő **Workspace** objektum létrehozásakor egy **Workspace** munkamenet indul el, és ezt követően az alkalmazásainkban már hivatkozhatunk erre a létrehozott objektumunkra.

Workspaces halmazból **Workspace** objektumot a **Delete** metódussal távolíthatunk el.

Delete

Szintaxis 1: *objektum*.**Delete**

Szintaxis 2: *rekordhalmaz*.**Delete**

Szintaxis 3: *objektumgyűjtemény*.**Delete** *objektumnév*

A metódus egy adott objektumot, egy nyitott objektum éppen aktuális rekordját vagy egy objektumgyűjtemény egy tárolt objektumát törli. A metódus szintaxisa a következő objektumazonosítókat és argumentumot tartalmazza:

- Az *objektum* azonosítója az **OLE** befogadó vezérlőelemen egy objektumot kije-

lölő objektumkifejezés.

- A *rekordhalmaz* azonosító a törlendő rekordot tartalmazó, nyitott dinamikus vagy tábla típusú **Recordset** (valamint az elavult **Dynaset** vagy **Table**) objektumot megadó objektumtípusként deklarált változó.
- Az *objektumgyűjtemény* azonosító egy objektumgyűjteményt azonosító objektumtípusú változó.
- Az *objektumnév* argumentum egy objektumgyűjteményben lévő objektum **Name** tulajdonságát tartalmazó **String**.

A **Delete** metódussal egy adott objektum, egy rekordkészletből az éppen aktuális rekord, egy halmazból egy tag, egy adatbázisból egy tárolt tábla, egy táblából egy tárolt mező vagy egy táblából egy tárolt index törölhető a következők szerint.

Az **OLE** befogadó vezérlőelemeken:

- A metódus az *objektum* azonosítójában adott objektumot törli, valamint felszabadítja a hozzá tartozó memóriaterületet, így lehetővé téve egy objektum explicit törlését. (Az objektum egy űrlap bezárásakor vagy az objektumnak egy új objektummal történő helyettesítése esetén automatikusan törlődik.)

A rekordhalmazokon:

- A törlendő rekordnak törlés előtt a rekordhalmaz éppen aktuális rekordjának kell lennie, különben kezelhető hiba következik be.
- A **Delete** metódus a tábla és a dinamikus típusú **Recordset**, (valamint az elavult **Dynaset** vagy **Table**) objektumokból eltávolítja, és elérhetetlenné teszi az aktuális rekordot. Ekkor az ugyan nem szerkeszthető vagy nem használható, azonban mindaddig aktuális rekord marad, amíg másik rekordra nem lépünk, ezután a törölt rekordot már nem tehetjük aktuális rekorddá. A **Recordset** (valamint az elavult **Table**) objektumban a törölt rekordra történő alárendelt hivatkozások érvénytelenek és hibát okoznak.
- Egy rekord visszaállítása a **Rollback** tranzakciós metódus használatával lehetséges.
- Ha az alaptábla egy lépcsőzetes, kaszkád törlési kapcsolatban elsődleges tábla, az aktuális rekord törlése egyben egy vagy több rekordot törlését is eredményezi egy idegen táblán.

Az objektumgyűjteményeken:

- A **Delete** metódussal törölhetjük az állandó objektumokat, azonban a csak memóriában tárolt **Databases**, **Recordsets** vagy **Workspaces** objektumgyűjtemények esetében egy nyitott vagy aktív objektumot a **Close** metódussal bezárva lehet eltávolítani.
- Egy tárolt objektum törlése azonnali, de a **Refresh** metódus bármely más, az

adatbázis szerkezetének változása által befolyásolt halmazon használható.
- Amikor egy **TableDef** objektumot a **TableDefs** halmazból törlünk, a tábla definíciója és adatai is törlődnek.

A következő táblázat első oszlopában szereplő objektumokat tartalmazzák a második oszlopban szereplő halmazok, a harmadik oszlopban pedig a **Delete** metódus törlésre vonatkozó korlátozásai szerepelnek.

objektum:	objektumgyűjtemény:	A **Delete** használhatósága:
Container	**Documents**	Soha
Database	**Containers**	Soha
Database	**Recordsets**	Soha, helyette a **Close** metódus használható
Database	**Properties**	Ha a tulajdonság felhasználó által definiált
DBEngine	**Workspaces**	Soha, helyette a **Close** metódus használható
Field	**Properties**	Ha a tulajdonság felhasználó által definiált
Index	**Fields**	Új, adatbázishoz még nem csatolt objektum esetén
Index	**Properties**	Ha a tulajdonság felhasználó által definiált
QueryDef	**Fields**	Soha
QueryDef	**Parameters**	Soha
QueryDef	**Properties**	Ha a tulajdonság felhasználó által definiált
Recordset	**Fields**	Soha
Relation	**Fields**	Új, adatbázishoz még nem csatolt objektum esetén
TableDef	**Fields**	Új, adatbázishoz még nem csatolt objektum esetén, vagy ha a **TableDef Updatable** tulajdonsága **True**
TableDef	**Indexes**	Új, adatbázishoz még nem csatolt objektum esetén, vagy ha a **TableDef Updatable** tulajdonsága **True**
TableDef	**Properties**	Ha a tulajdonság felhasználó által definiált
Workspace	**Databases**	Soha, helyette a **Close** metódus használható

Egy cikluson belül bármely, egynél több rekordot szerkesztő DAO (Data Access Object) metódus használatakor (**AddNew**, **Delete**, **Edit** vagy **Update**) a Microsoft Jet 3.0-s verzió a teljes ciklust egy implicit tranzakcióként kezeli, vagyis minden egyes frissítést nem szükségszerűen írhatjuk azonnal az adatbázisba, ami hatást gyakorolhat a hibakezelésre, az oldalzárolásra és a változásoknak a többi felhasználó általi láthatóságára. A változások azonnali adatbázisba írásához egy explicit tranzakción belül minden egyes **Edit...Update** blokkot be kell zárni.

DeleteQueryDef

Szintaxis:

Ez a **Database** objektumon használható, adott adatbázisból egy **QueryDef** objektumot törlő metódus a Visual Basic 4.0-s verziójában csak kompatibilitási okokból maradt

meg, helyette a **QueryDefs** objektumgyűjtemény **Delete** metódusának használata javasolt.

DoVerb

Szintaxis: *objektum*.**DoVerb** [(*módszer*)]

Ez az **OLE** befogadó vezérlőelemen használható metódus egy objektumot valamilyen műveletre nyit meg. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *módszer* argumentum az **OLE** befogadó vezérlőelemen belül egy objektum végrehajtási módját adja meg. Ha nincs megadva, az alapértelmezett módszer szerinti végrehajtás következik be. Az argumentum lehetséges értéke az összes objektum által támogatott standard módszerek egyike vagy az **ObjectVerbs** tulajdonságtömb egy indexe lehet.

Minden egyes objektum saját módszerkészletet támogat, de az összes objektum által támogatott — az **ObjectVerbs** tulajdonságtömbben nem szereplő — standard módszerek a következők:

Ha a szimbolikus konstans:	Értéke:	Hatása:
dbOLEPrimary	0	Az objektum alapértelmezett műveletét hajtja végre
dbOLEShow	-1	Az objektumot szerkesztésre aktiválja. Ha az objektumot létrehozó alkalmazás támogatja a helybeni aktiválást, az objektum az **OLE** befogadó vezérlőelemen belül aktiválódik.
dbOLEOpen	-2	Az objektumot egy különálló alkalmazási ablakban nyitja meg. Ha az objektumot létrehozó alkalmazás támogatja a helybeni aktiválást, az objektum a saját ablakán belül aktiválódik.
dbOLEHide	-3	A beágyazott objektumok esetén elrejti az objektumot létrehozó alkalmazást
dbOLEActivate	-4	Ha az objektum támogatja, az objektum helyben aktiválódik és láthatóvá teszi a felhasználói interfész-eszközöket. Ha az objektum nem támogatja a helybeni aktiválást, az objektum nem válik aktívvá és hiba következik be.
dbOLEInPlaceActivate	-5	Ha a felhasználó az **OLE** befogadó vezérlőelemre állítja a fókuszt, egy ablakot hoz létre az objektum számára és az objektumot szerkesztésre készíti elő.

dbOLEDiscardUndoState	-6	Ha az objektum nem támogatja az egyetlen egérkattintással történő aktiválást, hiba következik be. Akkor használatos, ha az objektum az alkalmazásai által okozott összes változást érvénytelenítő szerkesztésre aktivált.

Ha az **AutoActivate** tulajdonság értéke 2 (*Double-Click*), az **OLE** befogadó vezérlőelem a rajta történő dupla kattintással automatikusan aktiválja az aktuális objektumot.

Drag

Szintaxis: *objektum*.**Drag** [*művelet*]

A metódus a **CheckBox**, **ComboBox**, **CommandButton**, **DBGrid**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemeken a **Line**, **Menu**, **Shape**, **Timer** vagy **CommonDialog** kivételével az összes többi vezérlőelem egérrel történő mozgatásának, elhúzásának megkezdésére, befejezésére vagy érvénytelenítésére szolgál. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha nincs megadva, az objektum a **Drag** metódust tartalmazó eseményeljáráshoz tartozó objektum lesz.
- A feltételes *művelet* argumentum egy szimbolikus konstans vagy **Integer** típusú érték lehet, amely a végrehajtandó műveletet határozza meg. Ha az argumentum hiányzik, az alapértelmezett művelet az objektum elhúzásának megkezdése lesz.

A *művelet* argumentum lehetséges értékei a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	Hatása:
dbCancel	0	Törli az elhúzási műveletet
dbBeginDrag	1	Az *objektum* elhúzásának kezdete
dbEndDrag	2	Az *objektum* elhúzásának befejezése

Az elhúzási művelet vezérlésére a **Drag** metódus csak akkor szükséges, ha az objektum **DragMode** tulajdonságának *szám* argumentuma 0, azaz manuális mozgatás van beállítva, noha a tulajdonság *szám* argumentumának 1, azaz automatikus mozgatásra beállított vezérlőelemen is használható.

A vezérlőelem elhúzása alatt az egérmutató alakjának megváltoztatása a **DragIcon**, vagy **MousePointer** tulajdonság argumentumán keresztül lehetséges. A **MousePointer** csak akkor használható, ha nincs beállítva **DragIcon** tulajdonság.

A Visual Basic előző verzióiban a **Drag** aszinkron metódus volt, vagyis még akkor is újabb utasítások voltak meghívva, ha az elhúzási művelet még nem fejeződött be. A

4.0-s verzióban a metódus szinkronná vált, vagyis mindaddig nem hívódnak meg újabb utasítások, ameddig az elhúzási művelet be nem fejeződik.

Edit

Szintaxis: *rekordhalmaz*.**Edit**

A metódus egy már megnyitott *rekordhalmaz* nevű, csak eredményhalmaz és tábla típusú **Recordset** (valamint az elavult **Dynaset** vagy **Table**) objektumról az aktuális rekordot a másolatpufferbe történő bemásolással szerkesztésre, módosításra megnyitja.

A metódus használatakor az aktuális rekord mezőinek változásai a másolatpufferbe kerülnek, a szerkesztés eredményét pedig az **Update** metódus használatával véglegesíthetjük.

Egy rekord szerkesztéséhez először az **Edit** metódus használatával az aktuális rekord tartalmát a másolatpufferbe kell másolni, különben a mezőértékek módosítását vagy az **Update** használatát megkísérelve hibát kapunk.

Amikor a többfelhasználós környezetben a **Recordset** objektum **LockEdit** tulajdonságának értéke **True** (pesszimisztikus zárolás), a rekord az **Edit** használatától a frissítés befejeződéséig zárolt marad. Ha a **LockEdit** tulajdonság értéke **False** (optimisztikus zárolás), a rekord zárolttá válik és a frissítés előtti, szerkesztetlen rekorddal hasonlítódik össze. Ha a rekord az **Edit** metódus használata óta változott, az **Update** metódus kezelhető hibával (3197) zárul. A külső (ODBC és telepíthető ISAM) adatbázisokon mindig az optimisztikus zárolás használatos.

Az **Edit** használatának előfeltétele egy aktuális rekord léte. Ha ilyen nincs, vagy a rekordkészlet nem egy megnyitott, eredményhalmaz vagy tábla típusú **Recordset** (valamint az elavult **Dynaset** vagy **Table**) objektumra hivatkozik, hiba következik be. Az érvényes rekord az **Edit** végrehajtását követően is érvényes marad.

A rekordon végzett módosítások elvesznek:

- Ha az **Edit** vagy **AddNew** metódust használtuk és azután az **Update** használata nélkül másik rekordra léptünk át.
- Ha az **Edit** vagy **AddNew** metódust használtuk, és azután az **Update** használata nélkül ismét egy **Edit** vagy **AddNew** metódust hívunk meg.
- Ha a **Bookmark** tulajdonságot másik rekordra állítjuk.
- Ha a *rekordhalmaz* azonosítóban adott nevű objektumot az **Update** használata nélkül zárjuk be.

Az **Edit** használata a következő feltételek mellett okoz hibajelzést:

- Nincs érvényes rekord
- Az adatbázis vagy rekordhalmaz csak olvasható
- A rekordon belül nincs frissíthető mező
- Az adatbázis vagy rekordhalmaz egy másik felhasználónak volt megnyitva.

- Egy másik felhasználó már zárolta a rekordunkat tartalmazó oldalt

Egy cikluson belül bármely, egynél több rekordot szerkesztő DAO (Data Access Object) metódus használatakor (**AddNew**, **Delete**, **Edit** vagy **Update**) a Microsoft Jet 3.0-s verzió a teljes ciklust egy implicit tranzakcióként kezeli, vagyis minden egyes frissítést nem szükségszerűen írhatjuk azonnal az adatbázisba, ami hatást gyakorolhat a hibakezelésre, az oldalzárolásra és a változásoknak a többi felhasználó általi láthatóságára. A változások azonnali adatbázisba írásához egy explicit tranzakción belül minden egyes **Edit...Update** blokkot be kell zárni.

EndDoc

Szintaxis: *objektum*.**EndDoc**

Ez a **Printer** objektumon használható metódus lezárja a **Printer** objektumra (nyomtatóra) küldött nyomtatási műveletet, valamint felszabadítja a nyomtató eszközt vagy nyomtatási sort. A metódus szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az **EndDoc** metódusnak közvetlenül a **NewPage** utáni meghívása esetén nem következik be járulékos lapdobás.

Execute

Szintaxis 1: *adatbázis*.**Execute** *forrás* [, *beállítás*]

Szintaxis 2: *lekérdezésdefiníció*.**Execute** [*beállítás*]

Ez a **Database** és **QueryDef** objektumokon használatos metódus egy módosító lekérdezést futtat vagy a megadott **Database** objektumon egy SQL utasítást hajt vége. A metódus szintaxisa a következő objektumazonosítókat és argumentumokat tartalmazza:

- Az *adatbázis* azonosító annak a **Database** objektumnak a neve, amelyen a lekérdezés futni fog.
- A *forrás* argumentum az SQL utasítás vagy a **QueryDef** objektum.
- A feltételes *beállítás* argumentum egy szimbolikus konstans vagy **Integer** típusú érték lehet, amely a lekérdezés adatintegritási jellemzőit határozza meg.
- A *lekérdezésdefiníció* azonosító annak a **QueryDef** objektumnak a neve, amely **SQL** tulajdonságának értéke a végrehajtandó SQL utasítást meghatározza.

A *beállítás* argumentum lehetséges értékei a következők lehetnek:

Ha a szimbolikus konstans:	Hatása:
dbDenyWrite	Többfelhasználós környezetben a többi felhasználótól megtagadja az írási engedélyt.
dbInconsistent	Inkonzisztens frissítés. Alapértelmezett.

dbConsistent	Konzisztens frissítés.
dbSQLPassThrough	SQL átadás, vagyis az SQL utasítást egy ODBC adatbázisnak adja át feldolgozásra.
dbFailOnError	Ha hiba következik be, visszavonja a frissítést.
dbSeeChanges	Futásidejű hibát generál, ha más felhasználó az általunk szerkesztett adatokat módosítja.

Az **Execute** metódus csak módosító lekérdezéseken érvényes, más típusú lekérdezésen hibát okoz. Mivel a módosító lekérdezés nem ad vissza rekordot, az **Execute** sem ad vissza rekordhalmazt.

Szintaktikusan helyes SQL utasítást és megfelelő jogosultságot megadva az **Execute** még akkor is sikeres, ha nem egyetlen sort kell módosítani vagy törölni. A metódussal történő frissítés vagy törlő lekérdezés futtatásakor mindig a *beállítás* argumentum **dbFailOnError** értékét kell használni, amely kezelhető hibát generál, és minden sikeres változtatást visszavon, ha az érintett rekordok bármelyike zárolt és nem frissíthető vagy törölhető.

Különösen többfelhasználós környezetben érhetünk el jobb eredményt az **Execute** metódust egy tranzakcióba ágyazva. Az aktuális **Workspace** objektumon használjuk a **BeginTrans**, majd az **Execute** metódust, végül zárjuk le a tranzakciót a **CommitTrans** metódussal, ily módon a változásokat mentve és a lekérdezés futása során elhelyezett zárolásokat feloldva.

A legutolsó **Execute** metódus által érintett rekordok számának meghatározásához a **Database** vagy **QueryDef** objektum **RecordsAffected** tulajdonságát lehet használni.

A **dbConsistent** és **dbInconsistent** egyidejű használata esetén az alapértelmezett (**dbInconsistent**) érték lesz érvényes.

Az **Execute** nem ad vissza rekordhalmazt, de megengedi olyan beállítások használatát is, amelyek befolyásolják az SQL utasítás végrehajtási módját. Az **Execute** metódust rekordokat választó lekérdezésen használva hiba következik be.

ExecuteSQL

Szintaxis:

Ez a **Database** objektumokon használható, egy ODBC adatbázison egy lekérdezést SQL utasításként lekérdező metódus a Visual Basic 4.0-s verziójában csak kompatibilitási okokból maradt meg, helyette a pillanatfelvétel típusú **Recordset** objektumot külső ODBC adatbázis SQL lekérdezéseket tartalmazó **QueryDef** objektumokkal létrehozó **OpenRecordset**, illetve a **dbSQLPassThrough** beállítású **Execute** metódus használata javasolt.

FetchVerbs

Szintaxis: *objektum*.**FetchVerb**

Ez az **OLE** befogadó objektumon használható metódus az adott objektum által támogatott módszerlistát frissíti. A metódus szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A frissített módszerlistát az **ObjectVerbs** tulajdonság tartalmazza.

FieldSize

Szintaxis: *változónév* = *rekordhalmaz*!*mező*.**FieldSize**( )

A metódus egy **Field** objektumon használható, és egy **Recordset** objektum **Fields** halmazában a Memo vagy OLE objektum **Field** objektumának byte-számát adja meg. A metódus szintaxisa a következő objektumazonosítót tartalmazza:

- A *változónév* argumentum egy **Long** vagy **Variant** típusú változó neve.
- A *rekordhalmaz* azonosító a **Fields** halmazt tartalmazó **Recordset** (valamint az elavult **Dynaset**, **Snapshot** vagy **Table**) objektumot meghatározó objektum típusú változó.
- A *mező* azonosító egy **Field** objektum neve, amely **Type** tulajdonsága **dbMemo** (Memo), **dbLongBinary** (OLE objektum) vagy ezekkel egyenértékű.

Mivel egy OLE objektum mező vagy Memo mező mérete 64K-nál nagyobb lehet, beállítható, hogy a **FieldSize** által egy változónak visszaadott érték elegendően nagy legyen a **Long** típusú változó tárolásához. A nagy mezők kezeléséhez a **FieldSize** metódust az **AppendChunk** és **GetChunk** metódusokkal együtt használhatjuk. A Memo és LongBinary típusok kivételével a **Field** objektum méretét a **Size** tulajdonsággal lehet meghatározni.

FillCache

Szintaxis: *rekordhalmaz*.**FillCache** [*sorok* [, *kezdet*]]

A metódus csak eredményhalmaz típusú **Recordset** objektumon használható egy ODBC adatforrásból származó adatokat tartalmazó **Recordset** objektumhoz tarrtozó helyi gyorsítótár egészének vagy egy részének feltöltésére. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A *rekordhalmaz* azonosító egy ODBC adatforrásból létrehozott **Recordset** objektum, például egy csatolt táblát képviselő **TableDef** objektum vagy egy **TableDef** objektumból származtatott **QueryDef**.
- A feltételes *sorok* argumentum egy **Integer** érték, amely a gyorsítótárba töltendő

sorok számát adja meg. Ha az argumentum hiányzik, a sorok számát a **CacheSize** tulajdonság értéke határozza meg.

- A feltételes *kezdet* argumentum egy könyvjelzőt meghatározó karakterlánc, amely a gyorsítótár-feltöltés kezdő rekordját jelöli. Ha az argumentum hiányzik, a kezdeti rekordot a **CacheStart** tulajdonság értéke határozza meg.

A gyorsítótár használata javítja a távoli kiszolgáló adataihoz történő hozzáférést. A gyorsítótár a helyi memória egy része, amely a kiszolgálótól leggyakrabban kért adatokat tárolja, feltételezve, hogy azok az alkalmazás futása során a következőkben is szükségesek lesznek. A szükséges adatot a Microsoft Jet elsőként a gyorsítótárban keresi, a több időt igénylő kiszolgáló helyett. A gyorsítótárban csak az ODBC adatforrásból származó adatok tárolódnak.

Ha nem akarunk arra várni, hogy a gyorsítótár az adatkeresés folyamán a rekordokkal egyenként töltődjön fel, a **FillCache** metódussal explicit módon is feltölthetjük. Ez sokkal gyorsabb folyamat, mivel a **FillCache** metódus egyszerre több rekordot olvas be.

Mindegyik, **Recordset** objektumokon keresztül elérhető ODBC adatbázisnak lehet helyi gyorsítótára, amelyet egy távoli adatforrástól egy **Recordset** objektum megnyitásával, majd a **CacheSize** és **CacheSart** tulajdonságok beállításával lehet létrehozni. Ha a *sorok* és a *kezdet* argumentumok olyan rekordtartományt jelölnek meg, amely teljesen vagy részben kívül esik a **CacheSize** és **CacheSart** tulajdonságokkal megadottól, a kívül eső rész figyelmen kívül marad, és nem töltődik a helyi gyorsítótárba. Ha a **FillCache** több rekordot kíván, mint amennyi a távoli adatforrásban maradt, a fennmaradó adatok hibajelzés nélkül töltődnek be. A **FillCache** csak a gyorsítótárba még nem töltött adatokat olvassa be.

Megjegyzendő, hogy a gyorsítótárban található adatok nem tükrözik a forrásadatokban más felhasználók által végrehajtott változtatásokat, ezért a gyorsítótárba töltött adatok frissítéséhez a **Recordset** objektum **CacheSize** tulajdonságát 0 értékre kell beállítani, majd a tárméretet az eredeti értékre visszaállítva a **FillCache** metódussal az aktuális adatok visszatölthetők.

FindFirst

Szintaxis: *rekordhalmaz*.**FindFirst** *feltétel*

A metódus megkeresi, és aktuális rekorddá teszi egy eredményhalmaz vagy pillanatfelvétel típusú **Recordset** (valamint az elavult **Dynaset** vagy **Snapshot**) objektum rekordjai közül az adott feltételnek megfelelő első rekordot. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- A *rekordhalmaz* azonosítója egy létező eredményhalmaz vagy pillanatfelvétel típusú **Recordset** (valamint az elavult **Dynaset** vagy **Snapshot**) objektum neve.
- A *feltétel* argumentum a rekord kiválasztási követelményét megadó karakteres kifejezés, amely hasonló egy SQL utasítás **WHERE** záradékához, a **WHERE**

kulcsszó nélkül.

Ha a keresést mindegyik (nem csak a *feltétel* argumentumnak megfelelő) rekordra akarjuk kiterjeszteni, a **Move** (**MoveFirst**, **MoveLast**, **MoveNext** vagy **MovePrevious**) metódus használatával lehet rekordról rekordra lépni. A **Find** (**FindFirst**, **FindPrevious**, **FindNext** vagy **FindLast**) metódusok hatása nem azonos a **Move** metódusokéval, amelyek minden feltétel nélkül csak egyszerűen az első, utolsó, előző vagy a következő rekordot teszi aktuális rekorddá. A tábla típusú **Recordset** (valamint az elavult **Table**) objektumon egy rekord keresésére a **Seek** metódus használható.

Ha nem található a feltételnek megfelelő rekord, az aktuális rekord a **Recordset** első rekordja lesz, a **NoMatch** tulajdonság pedig **True** értékre változik.

Ha a *rekordhalmaz* egynél több, a feltételnek megfelelő rekordot tartalmaz, azok közül a sorrendben első lesz kiválasztva, a következőt pedig a **FindNext** metódussal lehet meghatározni. A **FindFirst** metódussal a keresés a rekordhalmaz elejétől a vége felé irányul.

Az aktuális rekord szerkesztésekor a következő rekordra lépés előtt az **Update** metódussal a változtatást érvényesíteni kell, különben azok figyelmeztetés nélkül elvesznek.

A **FindFirst** metódus sikeres végrehajtását a **NoMatch** tulajdonság vizsgálatával ellenőrizhetjük: ha annak értéke **False**, a keresés sikeres volt.

A metódust nem lehet a csak előre gördíthető pillanatfelvétel típusú **Recordset** objektumon használni.

A **FindFirst** metódus az ODBC hozzáférésű rekordhalmazokon nem elég hatékony, különösen nagyméretű rekordhalmazokon gyorsabb egy adott rekord keresése az **SQL** újrafogalmazásával.

Az US dátumforma (hónap-nap-év) dátumot tartalmazó mezők keresésére akkor is használható, ha a Microsoft Jet nem US verziójú, azonban ekkor a dátumot a **Format** függvénnyel konvertálni kell.

ODBC adatbázisokkal és nagy eredményhalmaz rekordhalmazokkal dolgozva a **FindFirst** metódust vagy a **Sort** vagy **Filter** tulajdonság használatát lassúnak találhatjuk. A keresést testre szabott **ORDER BY** vagy **WHERE** záradékú **SQL** lekérdezésekkel, paraméterlekérdezésekkel vagy adott indexelt rekordokat lekérdező **QueryDef** objektumok használatával lehet gyorsabbá tenni.

FindLast

Szintaxis: *rekordhalmaz*.**FindLast** *feltétel*

A metódus megkeresi és aktuális rekorddá teszi egy eredményhalmaz vagy pillanatfelvétel típusú **Recordset** (valamint az elavult **Dynaset** vagy **Snapshot**) objektum rekordjai közül az adott feltételnek megfelelő utolsó rekordot. A metódus szintaxisában használt argumentumok értelmezése értelemszerűen megegyezik a **FindFirst** metódusnál leírtakkal.

Ha a *rekordhalmaz* egynél több, a feltételnek megfelelő rekordot tartalmaz, azok közül a sorrendben utolsó lesz kiválasztva, az előzőt pedig a **FindPrevious** metódussal lehet meghatározni. A **FindLast** metódussal a keresés a rekordhalmaz végétől az eleje felé irányul. A keresés előtt a Microsoft Jet a rekordhalmazt — ha az még nem lenne feltöltve — teljesen feltölti.

A metódussal kapcsolatos további tudnivalók értelemszerűen megegyeznek a **FindFirst** metódusnál leírtakkal.

FindNext

Szintaxis: *rekordhalmaz*.**FindNext** *feltétel*

A metódus megkeresi, és aktuális rekorddá teszi egy eredményhalmaz vagy pillanatfelvétel típusú **Recordset** (valamint az elavult **Dynaset** vagy **Snapshot**) objektum rekordjai közül az aktuális rekord után következő, adott feltételnek megfelelő rekordot. A **FindNext** metódussal a keresés a rekordhalmaz vége felé irányul. A szintaxisban használt argumentumok értelmezése és a metódussal kapcsolatos további tudnivalók értelemszerűen megegyeznek a **FindFirst** metódusnál leírtakkal.

FindPrevious

Szintaxis: *rekordhalmaz*.**FindPrevious** *feltétel*

A metódus megkeresi és aktuális rekorddá teszi egy eredményhalmaz vagy pillanatfelvétel típusú **Recordset** (valamint az elavult **Dynaset** vagy **Snapshot**) objektum rekordjai közül az aktuális rekord előtti, adott feltételnek megfelelő rekordot. A **FindPrevious** metódussal a keresés a rekordhalmaz eleje felé irányul. A metódust nem lehet a csak előre gördíthető pillanatfelvétel típusú **Recordset** objektumon használni. A szintaxisban használt argumentumok értelmezése és a metódussal kapcsolatos további tudnivalók értelemszerűen megegyeznek a **FindFirst** metódusnál leírtakkal.

GetBookmark

Szintaxis: *objektum*.**GetBookmark** *érték*

Ez a **DBGrid** vezérlőelemen érvényes metódus a vezérlőelem aktuális sorához viszonyított sor könyvjelzőjét tartalmazó értéket ad vissza. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum egy **Long** típusú kifejezés, amely a **DBGrid** vezérlőelem aktuális sorához viszonyított sort jelöli ki.

Az *érték* argumentum lehetséges értékei a következő táblázat szerintiek lehetnek:

Ha az *érték*:	Hatása:
0	Az aktuális sor könyvjelzőjét (= `DBGrid1.Bookmark`) adja vissza.
1	Az aktuális sor utáni sor könyvjelzőjét adja vissza.
-1	Az aktuális sor előtti sor könyvjelzőjét adja vissza.
n	Az aktuális sor utáni *n*-edik (= `DBGrid1.Row + n`) sor könyvjelzőjét adja vissza.

A **GetBookmark** metódus által visszaadott érték eltérő lehet a **RowBookmark** metódussal kapott értéktől, mivel az aktuális sor nem lehet rejtett.

GetChunk

Szintaxis: *karakteres_változó* = *rekordhalmaz*!*mező*.**GetChunk**(*eltolás*, *byte_szám*)

A **Field** objektumon használható metódus egy Memo vagy OLE objektum **Field** objektum tartalmát vagy annak egy részét adja vissza egy **Recordset** (valamint az elavult **Dynaset**, **Snapshot** vagy **Table**) objektum **Fields** objektumgyűjteményében. A metódus szintaxisa a következő objektumazonosítókat és argumentumokat tartalmazza:

- A *karakteres_változó* argumentum egy **String** változó vagy **Variant** neve, amely a *mező* argumentum által megnevezett **Field** objektumból származó adatokat fogadja. Mivel a **String** változó mérete korlátozott, a *byte_szám* argumentum nem lehet 64K-nál nagyobb. Az ennél nagyobb mező csak részekben kapható meg.
- A *rekordhalmaz* azonosító egy objektumtípusú változó, amely a **Fields** objektumgyűjteményt tartalmazó **Recordset** (valamint az elavult **Dynaset**, **Snapshot** vagy **Table**) objektumot határozza meg.
- A *mező* azonosító annak a **Field** objektumnak a neve, amely **Type** tulajdonsága **dbMemo** (Memo), **dbLongBinary** (OLE objektum) vagy ezekkel egyenértékű beállítású.
- Az *eltolás* argumentum egy numerikus kifejezés, amely a másolás megkezdése előtt átugrandó byte-ok számát adja meg. Ha ez 0, a másolás a mező első byte-jától kezdődik.
- A *byte_szám* argumentum ugyancsak egy numerikus kifejezés, amely a visszaadandó byte-ok számát adja meg. Ha a *byte_szám* nagyobb is lenne a mező byte-jainak számánál, csak a mezőben ténylegesen található byte-okat kapjuk vissza. A 64 Kbyte-nál nagyobb értékű *byte_szám* még akkor is hibát okoz, ha a *karakteres_változó* **Variant**.

A **String** típusú *karakteres_változó* által 64K-ra korlátozott *byte_szám* miatt a nagyobb mezők csak részletekben olvashatók be, majd a részek az **AppendChunk** metódussal egyesíthetők. A visszakapott byte-ok számát a **Len** függvénnyel lehet meghatározni. Ha az *eltolás* vagy a *byte_szám* argumentum negatív, a **GetChunk** az érték komplemensét használja.

GetData

Szintaxis: *objektum*.**GetData**([*forma*])

Ez a **Clipboard** objektumon használható metódus a vágólapról egy grafikát ad vissza. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *forma* argumentum egy szimbolikus konstans vagy **Integer** típusú érték, amely a **Clipboard** objektum grafikus formáját határozza meg. Az argumentumot kötelező zárójelek között megadni. Ha értéke 0 vagy hiányzik, a metódus automatikusan a megfelelő formát használja.

Az argumentumban szereplő feltételes *forma* a következők valamelyike lehet:

Ha a szimbolikus konstans:	Értéke:	A visszaadott forma:
dbCFBitmap	2	Bitkép (.BMP)
dbCFMetafile	3	Metafájl (.WMF)
dbCFDIB	8	Eszközfüggetlen bitkép (DIB)
dbCFPalette	9	Színpaletta

Ha a vágólapon nem található az adott formának megfelelő grafika, semmit sem kapunk vissza. Ha csak egy színpaletta létezik a **Clipboard** objektumon, egy minimális méretű (1*1) eszközfüggetlen bitkép jön létre.

GetFormat

Szintaxis: *objektum*.**GetFormat**(*forma*)

Ez a **Clipboard** objektumon használható metódus egy **Boolean** értéket vissza, amely azt jelzi, hogy a vágólapon egy elem az adott formának felel-e meg. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *forma* argumentum egy szimbolikus konstans vagy **Integer** típusú érték, amely a **Clipboard** objektum formáját határozza meg. Az argumentumot kötelező zárójelek között megadni.

Az argumentumban szereplő *forma* a következők valamelyike lehet:

Ha a szimbolikus konstans:	Értéke:	A forma:
dbCFLink	HBF00	DDE párbeszéd információ
dbCFText	1	Szöveg
dbCFBitmap	2	Bitkép (.BMP)
dbCFMetafile	3	Metafájl (.WMF)

dbCFDIB	8	Eszközfüggetlen bitkép (DIB)
dbCFPalette	9	Színpaletta

A visszaadott érték **True**, ha a **Clipboard** objektum megfelel az adott formának, ellenkező esetben az eredmény **False** lesz.

A **dbCFDIB** és **dbCFBitmap** állandó szerinti formáknál a **Clipboard** objektumon a grafika megjelenítésekor bármilyen színpaletta lehet.

GetRows

Szintaxis: *tömb* = *rekordhalmaz*.**GetChunk**(*sorok_száma*)

A **Recordset** objektumon használható metódus az objektum több sorát egy tömbbe olvassa. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A *tömb* argumentum egy **Variant** típusú változó neve, amely a visszaadott adatokat tárolja.
- A *rekordhalmaz* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *sorok_száma* argumentum a visszaadandó sorok száma.

A metódust egy **Recordset** objektumból egy vagy több teljes rekordnak egy kétdimenziós tömbbe történő másolására használhatjuk. A tömb első indexe (dimenziója) a mezőt, a második pedig a sor számát adja meg.

A *sorok_száma* argumentumban a meglévőnél több sort megadva is csak a ténylegesen található sorokat kapjuk vissza. A visszakapott sorok számát az **UBound** tulajdonsággal lehet meghatározni.

A sorok számát az alkalmazás igényeinek megfelelően kell megválasztani, de azt a rendelkezésre álló memória mérete korlátozhatja.

A **GetRows** metódus használata után az aktuális rekord a következő kiolvasatlan sor lesz, vagyis a pozícionálási hatása megegyezik a **Move** *sorok_száma* metóduséval.

Ha egymást követő hívásokkal az összes sort ki akarjuk olvasni, az **EOF** függvénnyel tudjuk ellenőrizni a **Recordset** végének elérését.

A **GetRows** a kívántnál akkor ad vissza kevesebb sort, ha eléri a **Recordset** végét, vagy a kívánt tartományon belül nem tud egy sort kiolvasni. Ez utóbbi esetben a kiolvasás megáll a nem olvasható rekordnál, és az válik aktuális rekorddá, de nem következik be futásidejű hiba. Ilyen jelenség akkor következhet be, ha egy *rekordhalmaz* egy rekordját egy másik felhasználó törölte.

GetText

Szintaxis: *objektum*.**GetText**([*forma*])

Ez a **Clipboard** objektumon használható metódus egy szöveget tartalmazó karak-

terláncot ad vissza. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *forma* argumentum egy szimbolikus konstans vagy **Integer** típusú érték, amely a **Clipboard** objektum formáját határozza meg. Az argumentumot kötelező zárójelek között megadni.

Az argumentumban szereplő *forma* a következők valamelyike lehet:

Ha a szimbolikus konstans:	Értéke:	A forma:
dbCFLink	HBF00	DDE párbeszéd információ
dbCFText	1	Szöveg (Alapértelmezett)

A visszaadott érték üres ("") karakterlánc, ha a **Clipboard** objektumon nincs az adott formának megfelelő szöveg.

Hide

Szintaxis: [*objektum*].**Hide**

A **Form** (vagy **MDIForm**) objektumon használható metódussal az adott objektum, azaz űrlap rejtetté válik, azonban a memóriából nem törlődik. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá.

A **Form** elrejtéskor eltűnik a képernyőről, és **Visible** tulajdonságának értéke **False** lesz. A rejtett **Form** vezérlőelemei a felhasználó számára mindaddig nem érhetők el, amíg a rejtettséget okozó eseménykezelő eljárás utolsó utasítása is le nem futott. A futó Visual Basic alkalmazások, és az egyéb, vele DDE-n keresztül kapcsolatot tartó alkalmazások és **Timer** események az ilyen vezérlőelemekhez is hozzáférhetnek.

Érdemes megjegyezni, hogy egy még be nem töltött objektumot megadásával az automatikusan a memóriába töltődik, de a képernyőn nem jelenik meg, azaz rejtett állapotba kerül.

Idle

Szintaxis: **DBEngine.Idle [dbFreeLocks]**

A **DBEngine** objektumon használható metódus felfüggeszti az adatfeldolgozást, és engedélyezi a Microsoft Jet számára a felfüggesztett feladatok befejezését. Használatával lehetőséget adunk a háttérben futó, nagymennyiségű adatfeldolgozás miatt nem azonnali feladatok végrehajtására. Ez a többfelhasználós, többfeladatos környezetben gyakran előforduló eset, ahol nincs elegendő adatfeldolgozási háttéridő egy rekordhalmaz minden rekordjának aktuális állapotban tartására.

Rendszerint az olvasási zárolások eltávolítódnak, és az adatok a helyi eredményhalmaz típusú **Recordset** objektumokban csak akkor frissítődnek, amikor — az egérmozgást is beleértve — nem történik más művelet. Az **Idle** rendszeres használatával megfelelő idő biztosítható az adatbázis-készítő számára a háttérben futó feldolgozási feladatok elfogására a szükségtelen olvasási zárolások feloldásával. Ha a feltételes **dbFreeLocks** szimbolikus konstanst argumentumot is megadjuk, a feldolgozás az összes zárolás feloldásáig késleltetődik.

A metódus használata egyfelhasználós környezetben szükségtelen, hacsak egy alkalmazásnak nem több példánya fut. Az **Idle** metódus a többfelhasználós környezetben megnöveli a teljesítményt, miután az adatbázis-készítőt az adatoknak lemezre töltésére kényszerítve felszabadítja a memóriazárolásokat.

Insert

Szintaxis: *objektum*.**Insert**(*oszlopindex*)

Ez a **Columns** objektumgyűjteményen használható metódus a **DBGrid** vezérlőelem **Collumns** halmazának megadott pozíciójába egy oszlopot illeszt be, és az új **Column** objektumot adja vissza. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *oszlopindex* argumentum egy egész érték, amely az új **Column** objektum **Columns** halmazon belüli beillesztési pozícióját adja meg.

Az argumentumban szereplő *oszlopindex* a következő értékek valamelyike lehet:

Ha az *oszlopindex*:	A beillesztési pozíció:
0	Az új oszlop a bal oldali szélső pozícióba kerül.
Count	Ha az *oszlopindex* értéke azonos a **Count** tulajdonság értékével, az új oszlop a jobb oldali szélső pozícióba kerül.
n	Az új oszlop balról az *n*-edik lesz a **Columns** halmazban.

A beillesztett új oszlop **Visible** tulajdonsága **False** lesz, a többi tulajdonság pedig az alapértelmezett értéket veszi fel. Kezdetben az új oszlopok kötetlenek, mivel a **DataField** tulajdonság értéke üres ("") karakterlánc. A **Columns** halmaz **Count** tulajdonsága követi az oszlopok számának növekedését.

InsertObjDlg

Szintaxis: *objektum*.**InsertObjDlg**

A metódus **OLE** befogadó objektumon használható, és egy **Insert Object** párbeszédpanelt jelenít meg. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő

objektumkifejezés.

Futásidőben ezt a párbeszédpanelt azért jelenítjük meg, hogy a felhasználó egy csatolt vagy beágyazott objektumot hozhasson létre az objektumtípus és az objektumot szolgáltató alkalmazás kiválasztásával.

A párbeszédpanel használatával létrehozható objektum típusát (csatolt, beágyazott vagy mindkettő) az **OLETypeAlowed** tulajdonsággal határozhatjuk meg.

Egy új objektum létrehozásakor az osztálynévvel kapcsolatos alkalmazásnak (például EXCEL.EXE) megfelelő módon — az alkalmazás telepítő programjával — regisztráltnak kell lennie az operációs rendszerben.

Item

Szintaxis: *objektum*.**Item**(*index*)

Ez a **Collection** objektumon, valamint **Columns** és **Form** objektumgyűjteményen használható metódus egy halmaz pozícióval vagy kulccsal adott tagját adja vissza. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum egy, a halmazon belüli pozíciót vagy kulcsot megadó kifejezés. Ha numerikus kifejezés, értékének 1 és a halmaz **Count** tulajdonságának értéke között kell lennie, ha pedig karakteres kifejezés, a tagnak a halmazhoz adásakor használt **key** argumentummal kell egyeznie.

A halmaz egyik tagjához sem illő *index* argumentum hibát okoz. Mivel az **Item** metódus a halmazok alapértelmezett metódusa, a `Print MyCollection(1)` kifejezés egyenértékű a `Print MyCollection.Item(1)` kifejezéssel.

KillDoc

Szintaxis: *objektum*.**KillDoc**

A metódus **Printer** objektumon használható, és hatására az aktuális nyomtatás befejeződik. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Ha az operációs rendszerben a Print Manager fut, és a háttérben futó nyomtatás engedélyezett, a **KillDoc** metódus azonnal törli az aktuális nyomtatási feladatot, és a nyomtató nem kap további nyomtatandó adatot.

Ha nem a Print Manager kezeli a nyomtatási feladatot (a háttérben futó nyomtatás nem engedélyezett), az adatok vagy azok egy része a **KillDoc** metódus hatása előtt még a nyomtatóra kerülhet, a nyomtató vezérlése pedig a lehetőségek szerinti időben kerül alapállapotba, és fejezi be a nyomtatást.

Line

Szintaxis: [*objektum.*] **Line** [**Step**] [(*x1*, *y1*)] - [**Step**] (*x2*, *y2*) [, *szín*] [, **B**] [**F**]

Ezzel a **Form** és **Printer** objektumon, valamint **PictureBox** vezérlőelemen használható metódussal az argumentumban szereplő értékeknek megfelelő vonal vagy téglalap rajzolható. A szintaxis a következő objektumazonosítót és argumentumokat tartalmazza:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha nincs megadva, az *objektum* az éppen fókuszban lévő **Form** lesz.
- A feltételes **Step** kulcsszóval azt jelezhetjük, hogy rajzolás kezdőpontját vagy végpontját az *objektum* **CurrentX** és **CurrentY** tulajdonságokkal adott aktuális koordinátáihoz viszonyítjuk.
- A feltételes *x1* és *y1* **Single** típusú argumentumok a vonal vagy téglalap kezdőpontjának koordinátáit adják meg. Az *x* a vízszintes, az *y* pedig a függőleges koordinátát jelöli. A mértékegységet az *objektum* **ScaleMode** tulajdonsága határozza meg. Ha ez az argumentum hiányzik, a kezdőpont a **CurrentX** és **CurrentY** tulajdonságokkal adott aktuális koordinátájú pontban lesz.
- Az *x2* és *y2* **Single** típusú argumentumok a vonal végpontjának koordinátáit adják meg. A mértékegységet az *objektum* **ScaleMode** tulajdonsága határozza meg.
- A feltételes, **Long** típusú *szín* argumentum a rajzolt vonal RGB színét állítja be. A szín az **RGB( )** vagy a **QBColor( )** függvénnyel határozható meg. Hiányzó *szín* argumentum esetén a rajzszín a **ForeColor** tulajdonságának megfelelő lesz. A zárt alakzat (téglalap) belsejének színét az *objektum* **FillColor** és **FillStyle** tulajdonságai határozzák meg.
- A feltételes **B** beállítás használata esetén az *x1*, *y1* és az *x2*, *y2* pontok egy téglalap egymással átlós csúcspontjaiként értelmezettek.
- A csak **B** beállítással együtt használható **F** beállítás rajzszínnel kitöltött téglalapot eredményez, míg a csak **B** beállítású téglalap a **FillColor** és **FillStyle** tulajdonságok értékétől függő színű és kitöltöttségű lesz. Az alapértelmezett **FillStyle** érték esetén a téglalap átlátszó.

A rajzolás vonalvastagságát a **DrawWidth**, a háttér előtti megjelenés módját pedig a **DrawMode** és **DrawStyle** tulajdonságok értéke határozza meg.

Az utasítás végrehajtása után a **CurrentX** és **CurrentY** értékei az argumentumban végpontként adott koordinátaértékeket veszik fel.

LinkExecute

Szintaxis: *objektum*.**LinkExecute** *karakteres_kifejezés*

A metódus **Label**, **PictureBox** és **TextBox** vezérlőelemen használható, és DDE

kapcsolaton keresztül a forrásalkalmazás számára egy parancs-karakterláncot küld. A szintaxis a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *karakteres_kifejezés* argumentum a forrásalkalmazás által felismerhető utasítást tartalmazza.

A *karakteres_kifejezés* aktuális tartalma a forrásalkalmazástól függ, például a Microsoft Excel és Word for Windows a szögletes zárójelben adott makróparancsokat fogadja el. A metódus használata esetén érvényes kifejezéseket az egyes forrásalkalmazások dokumentációja tartalmazza.

LinkPoke

Szintaxis: *objektum*.**LinkPoke**

A metódus **Label**, **PictureBox** és **TextBox** vezérlőelemen használható, és a vezérlőelem tartalmát DDE kapcsolaton keresztül a célalkalmazás számára adja át. A szintaxisban az *objektum* azonosítója a DDE kapcsolatban érintett célalkalmazást kijelölő, annak nevét megadó objektumkifejezés.

Ha az *objektum* egy

- egy **Label**, a **LinkPoke** metódus a **Caption** tulajdonság,
- egy **PictureBox**, a **LinkPoke** metódus a **Picture** tulajdonság,
- egy **TextBox**, a **LinkPoke** metódus a **Text** tulajdonság

értékét továbbítja a forrásalkalmazáshoz.

Jellemzően a DDE kapcsolatban az adatcsere a forrásalkalmazástól (kiszolgálótól) a célalkalmazás (ügyfél) felé irányul, de a **LinkPoke** megengedi az ellentétes irányú információátvitelt is. A metódus adatot nem fogadó forrásalkalmazással történő használata hibát okoz.

LinkRequest

Szintaxis: *objektum*.**LinkRequest**

A metódus **Label**, **PictureBox** és **TextBox** vezérlőelemen használható, és a forrásalkalmazást a vezérlőelem tartalmának DDE kapcsolaton keresztül történő frissítésére kéri fel. A szintaxisban az *objektum* azonosítója a DDE kapcsolatban a célalkalmazást kijelölő, annak nevét megadó objektumkifejezés.

A metódus használatával a forrásalkalmazás az aktuális adatokat az *objektum* számára küldi, és ha az

- egy **Label**, a **Caption** tulajdonság,
- egy **PictureBox**, a **Picture** tulajdonság,

- egy **TextBox**, a **Text** tulajdonság

értékét frissíti.

A frissítési mód az objektum **LinkMode** tulajdonságának beállításától függően a következő lehet:

Ha a **LinkMode**:	Értéke:	A frissítési mód:
vbLinkAutomatic	1	*Automatic* — Az objektum adatainak frissítése automatikusan, a **LinkRequest** utasítás nélkül is megtörténik.
vbLinkManual	2	*Manual* — Az adatfrissítés csak a **LinkRequest** kiadásával hajtható végre.
vbLinkNotify	3	*Notify* — A kiszolgáló értesíti az ügyfelet, hogy **LinkNotify** eseményt okozó adatváltozás következett be, a felhasználónak pedig a **LinkRequest** metódus használatával kell az adatfrissítést elvégezni.

LinkSend

Szintaxis: *objektum*.**LinkSend**

A metódus **PictureBox** vezérlőelemen használható, és a vezérlőelem tartalmát DDE kapcsolaton keresztül a célalkalmazás számára adja át. A szintaxisban az *objektum* azonosítója egy, a DDE kapcsolatban forrásként szereplő objektumot kijelölő objektumkifejezés.

Amikor más alkalmazások automatikus kapcsolatot létesítenek az alkalmazásunk egy **Form** objektumával, a Visual Basic értesíti azokat a **Form** objektumon lévő **TextBox** vagy **Label** tartalmának változásáról, azonban a **PictureBox Picture** tulajdonságának változásakor a képek nagy adatmennyisége miatt célszerűtlen lenne egy-egy képpont változásakor a teljes tartalom frissítése, így a változásról szóló automatikus értesítés elmarad. A DDE ügyfélnek (kliensnek) a képtartalom változásáról történő explicit tájékoztatására a **LinkSend** metódust kell használni.

ListFields

Szintaxis:

Ez a **Dadatbase**, **Dynaset** és **Snapshot** objektumokon használható, egy adott rekordhalmazban mezőnként egy rekordot tartalmazó **Snapshot** objektumot létrehozó metódus a Visual Basic előző verzióival való kompatibilitás miatt maradt meg. A **ListFields** metódus helyett a listázandó mezőket tartalmazó objektum **Fields** halmazának felsorolása használható.

ListIndexes

Szintaxis:

Ez a **Table** objektumokon használható, egy adott tábla minden egyes indexében mezőnként egy rekordot tartalmazó **Snapshot** objektumot létrehozó metódus a Visual Basic előző verzióival való kompatibilitás miatt maradt meg. A **ListIndex** metódus helyett a listázandó indexeket tartalmazó objektum **Indexes** halmazának felsorolása használható.

ListParameters

Szintaxis:

Ez a **QueryDef** objektumokon használható, egy adott **QueryDef** objektum minden egyes argumentumában egy rekordot tartalmazó **Snapshot** objektumot létrehozó metódus a Visual Basic előző verzióival való kompatibilitás miatt maradt meg. A **ListParameter** metódus helyett a listázandó paramétereket tartalmazó objektum **Parameters** halmazának felsorolása használható.

ListTables

Szintaxis:

Ez a **Table** objektumokon használható, egy adott tábla minden egyes indexében mezőnként egy rekordot tartalmazó **Snapshot** objektumot létrehozó metódus a Visual Basic előző verzióival való kompatibilitás miatt maradt meg. A **ListIndex** metódus helyett a listázandó indexeket tartalmazó objektum **Indexes** halmazának felsorolása használható.

MakeReplica

Szintaxis: *adatbázis*. **MakeReplica(** *másolat* [, **dbRepMakeReadOnly**])

A metódus **Database** objektumon használható, és az adott fájlnév használatával az aktuális, többszörözhető adatbázisnak egy új másolati példányát hozza létre. A szintaxis a következő objektumazonosítót és argumentumot tartalmazza:

- Az *adatbázis* azonosítója egy létező, többszörözhető **Database** objektum neve.
- A *másolat* argumentum az új másolat számára adott elérési út és fájlnév. Ha ez már létező fájlnév, a metódus sikertelen lesz, és hibaüzenetet ad.

Ha a **dbRepMakeReadOnly** szimbolikus konstans is szerepel a szintaxisban, az újonnan létrehozott adatbázis többszörözhető elemei nem módosíthatók.

Move

Szintaxis 1: *rekordhalmaz*.**Move** *sorok* [, *kezdet*]

Szintaxis 2: [*objektum*].**Move** *bal* [, *felső* [, *szélesség* [, *magasság*]]]

A **Move** metódust kétféle objektumcsoporton, eltérő szintaxissal és eltérő hatással használhatjuk. A Szintaxis 1. szerinti **Move** metódus **Recordset** objektumokon érvényes, és hatására az objektum aktuális rekordjának pozíciója mozgatható.

A Szintaxis 1 a következő objektumazonosítót és argumentumokat tartalmazza:

- A *rekordhalmaz* azonosítója a mozgatandó pozíciójú rekordot tartalmazó **Recordset** objektum neve.
- A *sorok* argumentum egy **Long** érték, amely a mozgatandó sorok számát adja meg. Ha az argumentum pozitív, a mozgatás előre (a fájl vége) történik, ha pedig negatív, a sorok hátrafelé (a fájl eleje) mozgathatók.
- A feltételes *kezdet* argumentum egy, a mozgatás kezdetét megadó könyvjelzőt azonosító **String** érték. Ha ez az argumentum hiányzik, a mozgatás az aktuális rekordtól kezdődik.

Ha a **Move** metódus az aktuális rekordot az első rekord elé mozgatja, a rekord a fájlkezdet (BOF) pozícióba kerül. Ha a **Recordset** nem tartalmaz rekordot, és **BOF** tulajdonsága **True** értékű, a rekord visszafelé mozgatása kezelhető, futásidejű hibát okoz. Amennyiben a **BOF** vagy **EOF** valamelyike **True**, és a metódust érvénytelen könyvjelző nélkül akarjuk használni, kezelhető hiba következik be.

Hasonlóképpen, ha a **Move** metódus az aktuális rekordot az utolsó rekord után mozgatja, a rekord a fájlvég (EOF) pozícióba kerül. A lekérdezésen alapuló **Recordset** objektumoknál ilyen esetben a lekérdezés le fog futni az adott számú soron. Ha a **Recordset** nem tartalmaz rekordot, és **EOF** tulajdonsága **True** értékű, a rekord visszafelé mozgatása kezelhető, futásidejű hibát okoz.

A csak előre gördíthető pillanatfelvétel típusú **Recordset** objektumon használandó **Move** metódus szintaxisában a *sorok* értelemszerűen csak pozitív egész szám lehet, valamint nem használható a *kezdet* argumentum.

A rekordok mozgatására a **Move** metódus mellett a **MoveFirst**, **MoveLast**, **MoveNext** és **MovePrevious** metódusok is használhatók.

A **Move** metódus a Szintaxis 2 szerint **CheckBox**, **ComboBox**, **CommandButton**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **Line**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox**, és **VScrollBar** vezérlőelemeken, valamint **Form** és **MDIForm** objektumokon a vezérlőelem vagy az objektum mozgatására használható.

A Szintaxis 2 a következő objektumazonosítót és argumentumokat tartalmazza:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha nincs megadva, az *objektum* az éppen fókuszban lévő **Form** lesz.

- A *bal* argumentum egy **Single** érték, amely az *objektum* bal oldali élének x koordinátáját adja meg.
- A feltételes *felső* argumentum egy **Single** érték, amely az *objektum* felső élének y koordinátáját adja meg.
- A feltételes *szélesség* argumentum egy **Single** érték, amely az *objektum* szélességét adja meg.
- A feltételes *magasság* argumentum egy **Single** érték, amely az *objektum* magasságát adja meg.

Mint látható, a metódusban csak a *bal* argumentumot kötelező megadni, azonban a feltételes méretmódosító (*felső*, *szélesség*, *magasság*) argumentumok bármelyikének használata esetén kötelező a sorban előttük lévő argumentumok mindegyikét is megadni, különben az argumentum hatástalan lesz.

A metódusban használt argumentumok a **Form** és **MDIForm** objektumokon és **Frame** vezérlőelemen belüli vezérlőelemek esetén mindig twip egységben értelmezettek, a használt vonatkoztatási koordinátarendszer origója, vagyis (0,0) koordinátájú pontja pedig a bal felső sarokpont. Egy (logikai) cm = 567 twip.

Egy vezérlőelemnek a **Form** objektumon vagy **PictureBox** vezérlőelemen, valamint egy MDI-alűrlapnak az **MDIForm** objektumon belüli mozgatásakor a befogadó objektum koordinátarendszere lesz érvényes. A koordinátarendszer vagy mértékegység a programfejlesztés során a **ScaleMode** tulajdonságban állítható be, amelyet futásidőben a **Scale** metódussal lehet módosítani.

MoveFirst

Szintaxis: *rekordhalmaz*.**MoveFirst**

A metódus egy **Recordset** (valamint az elavult **Dynaset**, **Snapshot** vagy **Table**) objektumban, illetve **Recordsets** objektumgyűjteményben az első rekordot teszi aktuális rekorddá. A szintaxisban a *rekordhalmaz* azonosítója egy nyitott **Recordset** (valamint az elavult **Dynaset**, **Snapshot** vagy **Table**) objektum nevét tartalmazó objektumkifejezés.

A metódus használatával a rekordok közötti mozgás nincs feltételhez kötve. Ha feltételtől függő mozgásra van szükség, a **Find** (**FindFirst**, **FindLast**, **FindNext** vagy **FindPrevious**) metódusok, ha pedig feltétel nélkül adott számú rekorddal akarunk előre vagy hátrafelé lépni, a **Move** metódus használható. A tábla típusú **Recordset** (valamint az elavult **Table**) objektumon egy rekord keresésére a **Seek** metódus használható.

Az aktuális rekord szerkesztésekor a következő rekordra lépés előtt az **Update** metódussal a változtatást érvényesíteni kell, különben azok figyelmeztetés nélkül elvesznek.

A *rekordhalmaz* megnyitásakor annak első rekordja lesz az aktuális, és a **BOF** tulajdonság értéke **False**. Ha a *rekordhalmaz* nem tartalmaz rekordot, a **BOF** tulajdonsága **True** értékű, és természetesen nincs aktuális rekord.

Ha a *rekordhalmaz* egy tábla típusú **Recordset** (vagy az elavult **Table**) objektum, a

mozgás a tábla aktuális, az **Index** tulajdonsággal beállítható indexét követi. Ha nem állítunk be aktuális indexet, a visszakapott rekordok sorrendje határozatlan lesz.

A **MoveFirst** (és **MovePrevious**) metódust értelemszerűen nem lehet a csak előre gördíthető pillanatfelvétel típusú **Recordset** objektumon használni.

MoveLast

Szintaxis: *rekordhalmaz*.**MoveLast**

A metódus egy **Recordset** (valamint az elavult **Dynaset**, **Snapshot** vagy **Table**) objektumban, illetve **Recordsets** objektumgyűjteményben az utolsó rekordot teszi aktuális rekorddá.

A **MoveLast** metódust SQL lekérdezésen vagy **QueryDef** objektumon alapuló **Recordset** objektumon használva a lekérdezés végrehajtása befejeződik, és a **Recordset** objektum teljesen feltöltődik.

A metódus argumentumaira és használatára vonatkozó egyéb tudnivalók értelemszerűen azonosak a **MoveFirst** metódusnál leírtakkal.

MoveNext

Szintaxis: *rekordhalmaz*.**MoveNext**

A metódus egy **Recordset** (valamint az elavult **Dynaset**, **Snapshot** vagy **Table**) objektumban, illetve **Recordsets** objektumgyűjteményben a következő rekordot teszi aktuális rekorddá.

Ha a **MoveLast** metódust az utolsó aktuális rekordra alkalmazzuk, az **EOF** tulajdonság **True** értékre változik és a továbbiakban nem lesz aktuális rekord. Ilyenkor a metódus ismételt alkalmazása hibát okoz, de az **EOF** tulajdonság továbbra is **True** értékű marad.

A metódusra vonatkozó egyéb tudnivalók értelemszerűen azonosak a **MoveFirst** metódusnál leírtakkal.

MovePrevious

Szintaxis: *rekordhalmaz*.**MovePrevious**

A metódus egy **Recordset** (valamint az elavult **Dynaset**, **Snapshot** vagy **Table**) objektumban, illetve **Recordsets** objektumgyűjteményben az előző rekordot teszi aktuális rekorddá.

Ha a **MovePrevious** metódust a legelső aktuális rekordra alkalmazzuk, a **BOF** tulajdonság **True** értékre változik és a továbbiakban nem lesz aktuális rekord. Ilyenkor a metódus ismételt alkalmazása hibát okoz, de a **BOF** tulajdonság továbbra is **True** értékű marad.

A **MovePrevious** (és **MoveFirst**) metódust értelemszerűen nem lehet a csak előre gördíthető pillanatfelvétel típusú **Recordset** objektumon használni.

A metódusra vonatkozó egyéb tudnivalók értelemszerűen azonosak a **MoveFirst** metódusnál leírtakkal.

NewPage

Szintaxis: *objektum*.**NewPage**

A metódus **Printer** objektumon használható, és befejezi egy oldal éppen folyó nyomtatását, a következő oldalon a nyomtatási pozíciót visszaállítja a bal felső pozíciójába, valamint a **Printer** objektum **Page** tulajdonságának értékét eggyel megnöveli. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

NewPassword

Szintaxis: *felhasználói_objektum*.**NewPassword** *régi_jelszó*, *új_jelszó*

A **User** objektumon használható metódus egy létező felhasználói számla jelszavát módosítja. A szintaxis a következő objektumazonosítót és argumentumokat tartalmazza:

- A *felhasználói_objektum* azonosítója egy objektumtípusú változó, amely azt a **User** objektumot adja meg, amelynek **Password** tulajdonságát módosítani akarjuk.
- A *régi_jelszó* argumentum karakteres kifejezésében a **User** objektum **Password** tulajdonságának aktuális beállítását kell megadni.
- Az *új_jelszó* argumentum karakteres kifejezésében a **User** objektum **Password** tulajdonságának új beállítását kell megadni.

A jelszó megváltoztatásához előzőleg vagy egy bejelentkezett felhasználónak, vagy az Admin csoport tagjának kell lennünk.

A *felhasználói_objektum* azonosítójaként egy nem létező **User** objektum (**Users** objektumgyűjtemény-tag) megadása hibát okoz.

A *régi_jelszó* és *új_jelszó* karakteres kifejezések legfeljebb 14 karakter hosszúságúak lehetnek, és a 0 kivételével bármely ASCII karaktert tartalmazhatnak. A jelszóban a kis- és nagybetűk nem egyenértékűek. Az *új_jelszó* argumentumként üres karakterláncot ("") megadva a jelszó törlődik.

A **User** objektum **Password** tulajdonsága csak írható, az aktuális jelszót a felhasználó nem tudja kiolvasni.

OpenDatabase

Szintaxis: **Set** *adatbázis* = *munkaterület*.**OpenDatabase**(*dbnév* [, *kizárólagos* [, *csak_olvasható* [, *forrás*]]])

A **Workspace** objektumon használható metódus egy munkamenetben megnyit egy adatbázist, és a **Database** objektum egy hivatkozását adja vissza. A nyitott adatbázis automatikusan a **Databases** objektumgyűjteményhez adódik. A szintaxis objektumazonosítója és argumentumai a következők:

- Az *adatbázis* azonosítója egy **Database** objektumtípusú változó, amely a megnyitandó **Database** objektumot adja meg.
- A *munkaterület* azonosítója egy **Workspace** objektumtípusú változó, amely azt a létező **Workspace** objektumot adja meg, amely majd az adatbázist tartalmazza.
- A *dbnév* argumentum egy karakteres változó, amely egy létező adatbázisfájl vagy regisztrált ODBC adatforrás — esetleg a teljes elérési utat is tartalmazó — neve. Ha a fájlnévnek kiterjesztése is van, az is meg kell adni, például "C:\MYDB.MDB". Amennyiben a hálózat támogatja, a hálózati elérési utat például a "\\MYSERVER\MYSHARE\MYDIR\MYDB.MDB" alakban adhatjuk meg. Az argumentum használatakor a következőkre kell figyelemmel lenni:
  - Ha a *dbnév* más felhasználók által kizárólagos hozzáférésre már megnyitott adatbázisra, nem létező adatbázisra vagy érvényes ODBC adatforrásra hivatkozik, hiba következik be.
  - Ha a *dbnév* üres ("") karakterlánc és a forrás argumentum `"ODBC;"` , a párbeszédpanel megjeleníti az összes regisztrált ODBC adatforrás nevének listáját, amelyről a kívánt adatbázis kiválasztható.
- A feltételes *kizárólagos* argumentum egy **Boolean** típusú érték, amely akkor **True**, ha az adatbázis kizárólagos, nem megosztott hozzáférésre van megnyitva. Ha az argumentum hiányzik, az adatbázis az alapértelmezés (**False**) szerint megosztott hozzáférésre van megnyitva.
- A feltételes *csak_olvasható* argumentum egy **Boolean** típusú érték, amely akkor **True**, ha az adatbázis csak olvasható hozzáférésre van megnyitva. Ha az argumentum hiányzik, az adatbázis az alapértelmezés (**False**) szerint írási/olvasási hozzáférésre van megnyitva.
- A feltételes *forrás* argumentum az adatbázis megnyitásához az ODBC kapcsolati objektumazonosítót és argumentumokat képező karakteres kifejezés, amelyet csak a *kizárólagos* és *csak_olvasható* argumentumokkal együtt lehet használni. A kifejezés az adatbázis típusát, majd attól pontosvesszővel elválasztva a feltételes argumentumok ugyancsak pontosvesszővel elválasztott listáját tartalmazza, a szintaxisára vonatkozó további tudnivalók a **Connect** tulajdonság ismertetésénél találhatók.

Az adatbázis bezárásához és a **Database** objektumnak a **Databases** halmazból történő eltávolítására a **Close** metódus használható.

OpenQueryDef

Szintaxis:

Ez a **Database** objektumon használható, egy **QueryDef** objektumot későbbi szerkesztésre megnyitó metódus a Visual Basic 4.0-s verziójában csak kompatibilitási okokból maradt meg, helyette a **QueryDefs** objektumgyűjtemény, az **SQL** tulajdonság vagy a **QueryDef** objektum **Parameters** objektumgyűjtemény használata javasolt.

OpenRecordset

Szintaxis 1: **Set** *változó* = *adatbázis*.**OpenRecordset**(*forrás* [, *típus* [, *beállítás*]])

Szintaxis 2: **Set** *változó* = *rekordhalmaz*.**OpenRecordset**([*típus* [, *beállítás*]])

A **Database**, **QueryDef**, **Recordset** és **TableDef** objektumon használható metódus egy új **Recordset** objektumot hoz létre, és azt a **Recordsets** objektumgyűjteményhez adja. A nyitott adatbázis automatikusan a **Databases** objektumgyűjteményhez adódik. A szintaxisok objektum azonosítója és argumentumai a következők:

- A *változó* argumentum egy **Recordset** objektumtípusúként deklarált változó.
- Az *adatbázis* azonosítója egy létező, az új **Recordset** objektum létrehozásához használandó **Database** objektum neve.
- A *rekordhalmaz* azonosítója egy létező, az új **Recordset** objektum létrehozásához használandó **QueryDef**, **Recordset** vagy **TableDef** objektum neve, de az nem lehet csak előre gördíthető pillanatfelvétel típusú. Ha a *rekordhalmaz* egy eredményhalmaz vagy pillanatfelvétel típusú **Recordset**, a létrehozott új objektum típusa a *rekordhalmaz* azonosítójában szereplő **Recordset** típusával azonos. Ha a *rekordhalmaz* egy tábla típusú **Recordset**, az új objektum típusa egy eredményhalmaz típusú **Recordset** lesz.
- A *forrás* argumentum az új **Recordset** objektumhoz a rekordok forrását megadó **String**. A forrás lehet táblanév, lekérdezésnév vagy rekordokat visszaadó SQL utasítás. A tábla típusú **Recordset** esetén forrás csak táblanév lehet.
- A feltételes *típus* argumentum az új **Recordset** objektum típusát meghatározó szimbolikus konstans vagy **Integer** típusú érték. Ha hiányzik, az **OpenRecordset** metódus tábla típusú **Recordset** objektumot hoz létre, ha ez lehetséges. Hozzáfűzött tábla vagy lekérdezés megadása esetén az új objektum eredményhalmaz típusú **Recordset** lesz.
- A feltételes *beállítás* argumentum az új **Recordset** objektum jellemzőit előíró szimbolikus konstansok tetszőleges kombinációja lehet.

A *típus* argumentum a következő szimbolikus konstansok valamelyike lehet:

Ha a *típus*:	Az új **Recordset** típusa:
dbOpenTable	Tábla típusú **Recordset** objektum megnyitásához. Használata egy **QueryDef**, illetve eredményhalmaz vagy pillanatfelvétel típusú **Recordset** *rekordhalmaz*, illetve egy SQL utasítás vagy egy hozzáfűzött táblát képviselő **TableDef** *forrás* esetén kezelhető hibát okoz.
dbOpenDynaset	Eredményhalmaz típusú **Recordset** objektum megnyitásához
dbOpenSnapshot	Pillanatfelvétel típusú **Recordset** objektum megnyitásához

A *beállítás* argumentum a következő szimbolikus konstansok valamelyike lehet:

Ha a beállítás:	Az új **Recordset** jellemzője:
dbDenyWrite	Más felhasználók nem módosíthatják, csak rekordokat adhatnak hozzá.
dbDenyRead	Más felhasználók nem láthatják a rekordokat. (Csak tábla típusú **Recordset** esetén használható.)
dbReadOnly	A rekordokat csak megtekinthetjük, de más felhasználók módosíthatják is.
dbAppendOnly	Csak új rekordokat adhatunk hozzá. (Csak eredményhalmaz típusú **Recordset** esetén használható.)
dbInconsistent	Az inkonzisztens frissítés megengedett. Kizárja a **dbConsistent** konstans használatát. (Csak eredményhalmaz típusú **Recordset** esetén használható.)
dbConsistent	Csak konzisztens frissítés megengedett. Kizárja a **dbInconsistent** konstans használatát. (Csak eredményhalmaz típusú **Recordset** esetén használható.)
dbForwardOnly	A **Recordset** csak előre gördíthető pillanatfelvétel típusú lesz. Az így létrehozott objektumról nem lehet másolatot készíteni és a rekordok mozgatásához csak a **MoveNext** metódust támogatja. A beállítással javítható egy ODBC adatforrásból megnyitott **Recordset** objektumon keresztül történő átadás teljesítménye.
dbSQLPassThrough	Megkerüli a Microsoft Jet lekérdezés-feldolgozóját. Az **OpenRecordset** metódus forrás argumentumban adott lekérdezést feldolgozásra egy távoli ODBC kiszolgáló szintaxisellenőrzés vagy egyéb feldolgozás nélkül kapja meg. Ezzel a beállítással csak pillanatfelvétel típusú **Recordset** hozható létre, és ezt a típust kell a programkódban is megadni. Kötelező beállítás a csak egy adott ODBC kiszolgáló által támogatott SQL szintaxis használata esetén.

dbSeeChanges Az adat szerkesztése vagy törlése alatt más felhasználó által történt módosítás futásidejű hibát okoz, amely lekezelésekor a programunk frissítheti a mezőt és az új, módosított értékeket olvashatja vissza, illetve üzenetet küldhet az adatok változásáról.

A **Recordset** objektum **Close** metódussal történő bezárása azt automatikusan eltávolítja a **Recordsets** objektumgyűjteményből is.

Megjegyzendő, hogy a eredményhalmaz, pillanatfelvétel és tábla típusú **Recordset** objektumok eltérnek a Microsoft Jet 1.0-s és 1.1-s verzióiban megvalósított és már elavult **Dynaset**, **Snapshot** és **Table** objektumoktól, amelyek tulajdonságaik hasonlóak, mégsem teljesen "csereszabatosak", és számos Jet 3.0-s metódus és tulajdonság nem alkalmazható rájuk. Célszerű a régebbi alkalmazások programkódját az újabb **Recordset** objektumok használatára alkalmas alakra átírni.

OpenTable

Szintaxis:

Ez a **Database** objektumon használható, egy létező adatbázist megnyitó és egy **Table** objektumot létrehozó metódus a Visual Basic 4.0-s verziójában csak kompatibilitási okokból maradt meg, helyette a tábla típusú **Recordset** objektumot létrehozó **OpenRecordset** metódus használata javasolt. Megjegyzendő, hogy az **OpenRecordset** metódussal létrehozott **Table** objektum nem teljesen azonos az **OpenTable** metódussal létrehozottal, ahhoz kiegészítő funkcionális tulajdonságok járulnak.

PaintPicture

Szintaxis: [*objektum*].**PaintPicture** *kép*, *x1*, *y1* [, *szélesség1*] [, *magasság1*] [, *x2*, *y2*] [, *szélesség2*] [, *magasság2*] [, *kód*]

Ez a **Form** és **Print** objektumon, valamint **PictureBox** vezérlőelemen használható metódus egy grafikus fájl (.BMP, .WMF, .EMF, .ICO vagy .DIB) tartalmát az objektumra vagy vezérlőelemre rajzolja. A szintaxis objektumazonosítója és argumentumai a következők:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá.
- A *kép* argumentum az objektumba rajzolandó grafika forrása, amely egy **Form** vagy **PictureBox** objektum **Picture** tulajdonsága.
- Az *x1* és *y1* **Single** típusú argumentumok a rajzolandó képnek az objektumon érvényes célkoordinátáit adják meg. Az *x* a vízszintes, az *y* pedig a függőleges

koordinátát jelöli.

- A feltételes *szélesség1* **Single** típusú argumentum a rajzolandó kép célszélességét adja meg. Ha a kép célszélessége nagyobb vagy kisebb a forrásszélességnél (*szélesség2*), a kép ennek megfelelően nyomódik össze vagy húzódik szét. Ha a *szélesség1* hiányzik, az alapértelmezett érték a forrásszélesség lesz.
- A feltételes *magasság1* **Single** típusú argumentum a rajzolandó kép célmagasságát adja meg. Ha a kép célmagassága nagyobb vagy kisebb a forrásmagasságnál (*magasság2*), a kép ennek megfelelően nyomódik össze vagy húzódik szét. Ha a *szélesség1* hiányzik, az alapértelmezett érték a forrásmagasság lesz.
- A feltételes *x2* és *y2* **Single** típusú argumentumok a rajzolandó képen belüli kivágási tartomány koordinátáit adják meg. Ha hiányzanak, az alapértelmezett értékük 0 lesz.
- A feltételes *szélesség2* **Single** típusú argumentum a képen belüli kivágási tartomány forrásszélességét adja meg. Ha a *szélesség2* hiányzik, az alapértelmezett érték az eredeti forrásszélesség lesz.
- A feltételes *magasság2* **Single** típusú argumentum a képen belüli kivágási tartomány forrásmagasságát adja meg. Ha a *magasság2* hiányzik, az alapértelmezett érték az eredeti forrásmagasság lesz.
- A feltételes *kód* argumentum **Long** típusú érték vagy csak a bitképpel együtt használható, és egy bitenkénti (például **Not** vagy **Xor** logikai operátor) műveletet meghatározó programkód lehet, amelynek végrehajtása a rajzoláskor a képen történik.

A metódus minden argumentumának mértékegységét az *objektum* **ScaleMode** tulajdonsága határozza meg. A bitképet a *szélesség1* és/vagy *magasság1* argumentumokban negatív értéket használva lehet vízszintesen és/vagy függőlegesen átfordítani.

Az argumentumlista bármelyik feltételes tagja csak akkor érvényes, ha a szintaxisban előtte lévő összes argumentum is meg van adva. Az utolsó megadott argumentum utáni vesszőket nem kell kitenni.

Paste

Szintaxis: *objektum*.**Paste**

A metódus **OLE** befogadó vezérlőelemen használható adatoknak a rendszer vágólapjáról a vezérlőelemre másolására. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A metódus használatához az **OLETypeAllowed** tulajdonságot be kell állítani, majd ellenőrizni kell a **PasteOK** tulajdonság értékét. A másolás csak a **PasteOK** tulajdonság **True** értéke mellett lehetséges. Ha a másolás a **PasteOK** tulajdonság **True** értéke mellett sikertelen, azt az **OLE** befogadó vezérlőelem okozta a vezérlőelemen lévő összes objektum törlésével.

A **Paste** metódus végrehajtása után az **OLEType** tulajdonság értéke **vbOLELinked** (0) vagy **vbOLEEmbedded** (1), a sikertelen másolás esetén pedig **vbOLENone** (3) értékű lesz.

A metódust az Edit menü Paste parancsa támogatására lehet használni.

PasteSpecialDlg

Szintaxis: *objektum*.**PasteSpecialDlg**

A metódus **OLE** befogadó vezérlőelemen használható, és a Paste Special párbeszédpanelt jeleníti meg. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A metódust futásidőben a felhasználó számára egy objektumnak a rendszer vágólapjára történő másolását lehetővé tevő párbeszédpanel megjelenítésére használhatjuk. A párbeszédpanel számos beállítási lehetőséget ad a felhasználó számára, beleértve a csatolt vagy beágyazott objektumok másolását is.

A használathoz az **OLETypeAllowed** tulajdonsággal lehet meghatározni a Paste Special párbeszédpanellel létrehozható objektum (*Linked*, *Embedded* vagy *Either*) típusát.

A másolás csak a **PasteOK** tulajdonság **True** értéke mellett lehetséges. Ha a másolás a **PasteOK** tulajdonság **True** értéke mellett sikertelen, azt az **OLE** befogadó vezérlőelem okozta a vezérlőelemen lévő összes objektum törlésével.

Point

Szintaxis: [*objektum*].**Point**(*x*, *y*)

Ezzel metódussal egy **Form** objektum vagy **PictureBox** vezérlőelem meghatározott pontjában érvényes RGB színérték kérdezhető le. A szintaxis objektumazonosítója és argumentumai a következők:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá.
- Az *x* és *y* **Single** típusú argumentumok a kijelölendő pont koordinátáit adják meg. Az *x* a vízszintes, az *y* pedig a függőleges koordinátát jelöli. Ha az adott pont az *objektum* területén kívül esik, a **Point** metódus -1 értéket ad vissza. Az argumentumokat zárójelben kell megadni. A metódus *x* és *y* argumentumainak mértékegységét az *objektum* **ScaleMode** tulajdonsága határozza meg.

PopupMenu

Szintaxis: [*objektum*].**PopupMenu** *menünév* [, *jelzők* [, *x*, *y* [, *félkövér*]]]

Ez a metódus egy **Form** vagy **MDIForm** objektumon egy előugró menüt jelenít meg az aktuális egérmutató vagy az argumentumban adott koordináták helyén. A szintaxis objektumazonosítója és argumentumai a következők:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá.
- A *menünév* argumentumban a megjelenítendő előugró menü nevét kell megadni. A menünek legalább egy almenüt kell tartalmaznia.
- A feltételes *jelzők* argumentum egy szimbolikus konstans vagy **Integer** típusú érték, amely az előugró menü helyét és viselkedését határozza meg. A *jelző* a Windows 3.0-s vagy előbbi verziókban hatástalan.
- A feltételes *x* és *y* **Single** típusú argumentumok az előugró menü megjelenítési helyének az adott objektumhoz viszonyított koordinátáit adják meg. Az *x* a vízszintes, az *y* pedig a függőleges koordinátát jelöli. A metódus *x* és *y* argumentumainak mértékegységét az *objektum* **ScaleMode** tulajdonsága határozza meg. Az *x* és *y* argumentumok hiányában az egérmutató koordinátái lesznek az alapértelmezettek.
- A feltételes *félkövér* argumentum az előugró menü félkövér karakterekkel megjelenítendő vezérlőelemének nevét adja meg. Ha az argumentum hiányzik, a menüben nem lesz ilyen megjelenésű vezérlőelem. Ez az argumentum csak Windows 95 alatt futó alkalmazások esetén érvényes, a Windows 16-bites verziójában vagy a Windows NT 3.51-s vagy azelőtti verzióiban az alkalmazás ezt az argumentumot figyelmen kívül hagyja.

A szintaxis feltételes *jelzők* argumentumában használható szimbolikus konstans vagy érték a következő két táblázatban szereplők valamelyike lehet:

Ha a szimbolikus konstans:	Értéke:	A menü helye:
vbPopupMenuLeftAlign	0	Az előugró menü bal oldala az *x* koordinátának megfelelő lesz. (Alapértelmezett)
vbPopupMenuCenterAlign	4	Az előugró menü az *x* koordinátára központosított lesz.
vbPopupMenuRightAlign	8	Az előugró menü jobb oldala az *x* koordinátának megfelelő lesz

Ha a szimbolikus konstans:	Értéke:	A menü viselkedése:
vbPopupMenuLeftButton	0	Az előugró menü eleme csak a bal egérgombbal kattintásra reagál. (Alapértelmezett)

vbPopupMenuRightButton	2	Az előugró menü eleme a jobb vagy bal egérgombbal kattintásra is reagál.

A két csoportból egy-egy beállítását az értékek összegének megadásával, illetve a szimbolikus konstansok **Or** művelettel való összekapcsolásával kombinálhatjuk. Ha a menüelemet csak a jobb egérgombbal kívánjuk kiválasztani, erre a **MouseDown** eseményt lehet használni.

Futásidőben a **PopupMenu** metódus mindaddig felfüggeszti a programfutást, ameddig a menüből egy elemet nem jelölünk ki, vagy a menüt nem érvénytelenítjük. A menü egy elemét kiválasztva elsőként az ahhoz tartozó **Click** esemény végrehajtása történik meg, majd a programfutás a **PopupMenu** metódus utáni utasítással folytatódik.

Egyidőben csak egy menü jeleníthető meg, tehát egy legördülő vagy előugró menü megjelenését követően egy második menüt aktiváló **PopupMenu** metódus érvénytelen lesz.

Az olyan előugró menü, amely a menüsorban a **Visibe** tulajdonságának **False** értéke miatt láthatatlan, a Menu Editor ablakban mégis látható lesz, mivel a menüszerkesztő figyelmen kívül hagyja ezt a tulajdonságot.

Print

Szintaxis: [*objektum*].**Print** { **Spc**(*n*) | **Tab**(*n*) } [*kifejezéslista*] [*karakterpozíció*]

A **Print#** utasításhoz hasonló felépítésű metódus használatával a **Debug** ablak Immediate ablaktáblájába, a **Form** és **Printer** objektumba vagy **PictureBox** vezérlőelembe az érvényes színnel és karaktertípussal szöveget lehet írni. A szintaxis argumentumai a következők:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá.
- A feltételes **Spc**(*n*) használatával *n*-számú szóközt lehet a nyomtatandó szövegbe illeszteni.
- A feltételes **Tab**(*n*) használatával a beillesztési pontot az *n*-ik abszolút oszlopba lehet beállítani. Ha *n* értéke 0, a beillesztési pont a következő nyomtatási zóna elejére kerül.
- A feltételes *kifejezéslista* argumentum a nyomtatandó numerikus vagy karakteres kifejezések egymástól szóközzel vagy pontosvesszővel elválasztott listáját tartalmazza. Ha az argumentum **Empty** érték, semmi sem íródik ki, de ha **Null**, a kiírt szöveg a `Null` lesz.
- A feltételes *karakterpozíció* argumentum a következő karakter beillesztési pontját adja meg. Ha az argumentum ";", a beillesztési pont közvetlenül az utolsó kiírt karakter utáni lesz. A **Tab**(*n*) használatával a következő nyomtatandó ka-

rakter az *n*-ik abszolút oszlopba (0 érték esetén a következő nyomtatási zóna elejére) kerül. Ha az argumentum hiányzik, a következő karakter a következő sorba nyomtatódik.

A **Debug** objektumon az Immediate ablaktáblára minden adat megfelelően formázottan kerül kiírásra. A **Print** metódust **Printer** objektummal használva a grafikus kép a lap aljára, a szöveg pedig a következő lapra kerül. Ha a *kifejezéslista* **Boolean** típusú adatot tartalmaz, a `True` vagy `False` szó íródik ki. A **Date** típusú adatok a rendszerünkben érvényes rövid alakban íródnak ki, míg a hibaadatok kiírása `Error` *`hibaszám`* alakú lesz.

Mivel a **Print** metódus általában arányos szélességű karaktereket használ, a karakterszám és az elfoglalt sorhossz között nincs szoros összefüggés, hiszen például a W karakter több helyet foglal el, mint az l. Az űrlapok és hasonló táblázatok esetén érdemes ellenőrizni, hogy a rendelkezésre álló hely elegendő-e a széles karakterek beírásához is, vagy rögzített szélességű (például Courier) karaktertípust kell használni.

PrintForm

Szintaxis: [*objektum*].**PrintForm**

A metódus egy **Form** objektumról minden látható objektumot és képet bitenként a nyomtatóra küld. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá.

A **PrintForm** metódus nem csak minden látható objektumot és képet, de a futásidőben a **Form** objektumhoz vagy **PictureBox** vezérlőelemhez adott grafikát is nyomtatja, ha a grafika rajzolásakor az **AutoRedraw** tulajdonság **True**.

A **PrintForm** metódus által használt nyomtatót az operációs rendszer Control Panel beállítása határozza meg.

PSet

Szintaxis: [*objektum*].**PSet** [**Step**] (*x*, *y*) [, *szín*]

Ez a metódus a **Form** és **Printer** objektumon, valamint **PictureBox** vezérlőelemen egy adott pont RGB színét állítja be. A szintaxis argumentumai a következők:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá.
- A feltételes **Step** kulcsszó azt jelzi, hogy a koordinátákat az aktuális grafika **CurrentX** és **CurrentY** tulajdonságokkal adott koordinátáihoz viszonyítjuk.
- Az *x* és *y* **Single** típusú argumentumok a beállítandó pont koordinátáit adják

meg. Az *x* a vízszintes, az *y* pedig a függőleges koordinátát jelöli.

- A feltételes *szín* **Long** típusú argumentumban a pont RGB színét adhatjuk meg. Ha hiányzik, az aktuális **ForeColor** tulajdonság értéke lesz érvényes. A **BackColor** színét beállítva az adott pont rejtetté tehető. A szín megadásához az **RGB** vagy **QBColor** függvényt használhatjuk.

A rajzolt pont nagysága a **DrawWidth** tulajdonság értékétől függ, ha ez 1, a pont csak egy képpontnyi, ha ennél nagyobb, az adott koordinátára központosított grafikus alakzat lesz. A pontrajzolás módját a **DrawMode** és **DrawStyle** tulajdonságok határozzák meg.

A **PSet** végrehajtása után a **CurrentX** és **CurrentY** értéke az argumentumban adott *x* és *y* koordinátaértékekkel egyező lesz.

Raise

Szintaxis: *objektum*.**Raise**(***Number*** [, ***Source*** [, ***Description*** [, ***HelpFile*** [, ***HelpContext***]]]])

Ez a metódus az **Err** objektumon futásidejű hibát hoz létre. A szintaxis névvel hivatkozott argumentumai a következők:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A ***Number*** **Long** típusú argumentum, és a hiba természetét határozza meg. A Visual Basicben a belső és a felhasználó által definiált hibák azonosítószámainak egyaránt a 0 — 65535 közötti tartományba kell esnie. Egy OLE Automatizmus objektumban a saját hibaszám beállításához a hibaszámot a **vbObjectError** szimbolikus konstanshoz kell hozzáadni, például a saját definiálású 1050 hibaszám esetén a **vbObjectError**+1050 értékét kell adni a **Number** tulajdonságnak.
- A feltételes ***Source*** argumentum egy karakteres kifejezés, amely a hibát eredetileg generáló alkalmazás vagy objektum neve. Egy OLE Automatizmus objektumban a *projekt.osztály* alak használatával lehet ezt a tulajdonságot beállítani. Ha semmi sincs megadva, az aktuális Visual Basic projekt programazonosítója lesz érvényes.
- A feltételes ***Description*** argumentum egy, a hibát leíró karakteres kifejezés. Ha hiányzik, a **Number** tulajdonság értéke kerül vizsgálat alá. Amennyiben az a Visual Basic futásidejű hibaszámlistájában szerepelhet, az **Error** függvény által visszaadott érték kerül a ***Description*** argumentumba. Ha nem lenne a **Number** értékének megfelelő Visual Basic hibaszám, az `"Application-defined or object-defined error"` kerül az argumentumba.

- A feltételes ***HelpFile*** argumentum az adott hibához tartozó súgófájl teljes alakú elérési útja. Ha az argumentum hiányzik, a Visual Basic súgó fájl teljes (meghajtót, könyvtárat és fájlnevet is tartalmazó) elérési útja lesz érvényes.
- A feltételes ***HelpContext*** argumentum a hiba súgóját tartalmazó fájlon belül a hiba témakörét kijelölő azonosító. Ha nincs megadva az argumentum, az azonosító a **Number** tulajdonságnak megfelelő lesz.

A **Raise** metódus futásidejű hibákat hoz létre, és az **Error** utasítás helyett használható. Mint látható, a ***Number*** kivételével mindegyik argumentum feltételes, azonban a metódusnak egyes argumentumok nélküli használatakor a hibához az **Err** objektum szolgáltat bizonyos értékeket. Ez különösen az OLE Automatizmus objektumok írásakor hasznos, mivel az **Err** a programozónak és a felhasználónak jóval több információt ad, mintha az **Error** utasítással történne a hiba generálása.

ReadFromFile

Szintaxis: *objektum*.**ReadFromFile** *fájlszám*

Ez a metódus **OLE** befogadó objektumon használható, és a **SaveToFile** metódussal létrehozott adatfájlból tölt be egy objektumot. A szintaxis argumentumai a következők:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *fájlszám* argumentum a betöltendő objektumot tartalmazó nyitott, bináris fájlt kijelölő numerikus kifejezés.

Objektumot fájlba a **SaveToFile** vagy **SaveToOle1File** metódussal lehet menteni.

Rebind

Szintaxis: *objektum*.**Rebind**

A metódus egy **DBGrid** vezérlőelemen újra létrehozza a vezérlőelem tulajdonságait és oszlopait, valamint ugyanazokat a műveleteket hajtja végre, amelyek a vezérlőelem **DataSource** tulajdonságának beállításakor is bekövetkeznek, azaz újra beállítja az oszlopokat, fejléceket és az aktuális **Data** vezérlőelem tulajdonságain alapuló tulajdonságokat.

Refill

Szintaxis: *objektum*.**Refill**

A metódus újrarajzol egy **DBCombo** vagy **DBList** vezérlőelemet, valamint azon újra létrehozza a listát. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A metódus nem azonos a standard **Refresh** metódussal, amely csak egy **Repaint** eseményt eredményez.

Refresh

Szintaxis 1: *objektumgyűjtemény*.**Refresh**

Szintaxis 2: *objektum*.**Refresh**

A **Refresh** metódus két különböző szintaxisa eltérő elemeken érvényes, és természetszerűleg eltérő funkciót lát el.

A Szintaxis 1. szerinti metódus **Containers**, **Documents**, **Fields**, **Groups**, **Indexes**, **Parameters**, **Properties**, **QueryDefs**, **Relations**, **TableDefs** és **Users** objektumgyűjteményeken, illetve **Data** vezérlőelemen érvényes, és a halmazban lévő objektumot az aktuális adatbázisnak megfelelően frissíti, továbbá a **Recordset** objektummal vagy **Data** vezérlőelemmel létrehozott adatstruktúrák bezárására és újraépítésére is használható.

A **Refresh** metódust elsősorban többfelhasználós környezetben használhatjuk, amelyben más felhasználók módosíthatják az adatbázist. Az adatbázis változásával közvetetten érintett objektumgyűjtemények esetén is szükség lehet a metódus használatára.

Az objektumgyűjtemények az első hivatkozás alkalmával feltöltődnek objektumokkal és nem automatikusan tükrözik a többi felhasználó által végzett változtatásokat. Az ilyen halmazok adott objektumának egy alkalmazás által történő használata előtt a **Refresh** metódussal a halmazt frissíteni szükséges.

A metódus **Data** vezérlőelemen (ha a **DatabaseName**, **ReadOnly**, **Exclusive** vagy **Connect** tulajdonságok változtak) adatbázis megnyitására vagy újranyitására, továbbá a vezérlőelem **Recordset** tulajdonságára hivatkozó **Recordset** objektum újraépítésére használható.

A metódus nem állandó (például **Databases**, **Recordsets** vagy **Workspaces**) objektumgyűjteményeken nem érvényes.

A Microsoft Jet által a **QueryDef**, **Recordset** vagy **TableDef** objektum **Fields** halmazában a **Field** objektumokhoz használt pozíció meghatározásához az egyes **Field** objektumok **OrdinalPosition** tulajdonsága alkalmas. A halmazon belül egy **Field** objektum **OrdinalPosition** tulajdonságának változását követően használt **Refresh** metódus az objektumok rendjének megváltozását eredményezheti.

A **Refresh** metódus a Szintaxis 2 szerint **CheckBox**, **ComboBox**, **CommandButton**, **Data**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **HScrollBar**, **Image**, **Label**, **Line**, **ListBox**, **Menu**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox**, és **VScrollBar** vezérlőelemeken, valamint **Form** objektumokon teljes újrarajzolást hajt végre. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Általában egy űrlap vagy vezérlőelem újrarajzolása automatikusan történik, de előfordulhat, hogy azonnali frissítésre van szükségünk. A metódust akkor használhatjuk, ha

azt kívánjuk, hogy egy űrlap jelenjen meg a következő űrlap betöltése alatt, vagy egy fájllista (például **FileListBox** vezérlőelem), vagy egy **Data** vezérlőelem-adatstrukúra azonnal frissítődjön.

A **Refresh** metódust egy **Data** vezérlőelemen a **DatabaseName**, **ReadOnly**, **Exclusive** vagy **Connect** tulajdonság értékének megváltozásakor az adatbázis megnyitására vagy újranyitására, továbbá a eredményhalmaz halmaznak a vezérlőelem **Recordset** tulajdonságában történő újraépítésére is alkalmas.

A metódus **Menu** vagy **Timer** vezérlőelemen és **MDIForm** objektumon nem használható, de MDI-alablakokon igen.

RefreshLink

Szintaxis: *tábladefiníció*.**RefreshLink**

A metódus egy hozzáfűzött táblához frissíti a kapcsolatinformációkat.

A szintaxisban a *tábladefiníció* argumentum azt a hozzáfűzött táblát meghatározó **TableDef** objektumot adja meg, amelynek adatait frissíteni akarjuk.

A hozzáfűzött tábla kapcsolatleíró információinak módosítása a megfelelő **TableDef** objektum **Connect** tulajdonságának újbóli beállításával, majd az információ **RefreshLink** metódussal történő frissítésével lehetséges. Az így történő frissítés nem módosítja a hozzáfűzött tábla tulajdonságait és a **Relation** objektumokat.

A kapcsolatleíró információt a hozzáfűzött táblát meghatározó **TableDef** objektummal kapcsolatos minden objektumgyűjteményre a **RefreshLink** metódus minden egyes halmazra történő alkalmazásával lehet tenni.

RegisterDatabase

Szintaxis: **DBEngine.RegisterDatabase** *adatbázisnév, vezérlő, panel, attribútumok*

Ez a metódus **DBEngine** objektumon használható, és egy ODBC adatforráshoz tartozó kapcsolatleíró információkat helyez el az ODBC.INI fájlba. (Ezekre az információkra az ODBC vezérlőnek van szüksége, amikor a Microsoft Jet egy munkamenetben az adatforrást megnyitja). A szintaxis argumentumai a következők:

- Az *adatbázisnév* argumentum egy karakteres kifejezés, amely azonos az **OpenDatabase** metódusban használt névvel, és az adatbázisra vonatkozó leíró információblokkra hivatkozik, amely például egy távoli ODBC adatbázisnál a kiszolgáló neve lehet.
- A *vezérlő* argumentum az ODBC vezérlő nevét tartalmazó karakteres kifejezés, de ez nem az ODBC DDL-fájl neve, így például az SQL Server a vezérlő, az SQLSRVR.DLL pedig a DDL neve. A metódus használata előtt az ODBC-nek és a megfelelő vezérlőnek már telepítettnek kell lennie.
- A *panel* argumentum egy numerikus kifejezés, amellyel a vezérlőfüggő informá-

ciókat kérő ODBC vezérlő párbeszédpanel megjelenítését írhatjuk elő. Ha a kifejezés értéke **True**, a párbeszédpanel megjelenik, ha pedig **False**, nem. Ha a **True** érték esetén nem minden szükséges vezérlőfüggő információ áll rendelkezésre, a párbeszédpanel valamilyen, esetleg nem standard megjelenésű lesz.

- Az *attribútumok* argumentum egy karakteres kifejezés, amely az ODBC.INI fájlhoz adandó kulcsszavak kocsivissza-soremelés, azaz a **Chr**(13) **& Chr**(10) karakterekkel elválasztott listája.

Ha a **RegisterDatabase** metódus használatakor az adatbázis már regisztrált (vagyis a kapcsolatleíró információk már az ODBC.INI fájlban szerepelnek), a kapcsolatleíró információk frissítődnek.

A 32-bites telepítésnél a legtöbb ODBC regisztrációs információ az ODBC.INI fájl helyett a Windows regisztrációs adatbázisába kerül.

Ha a metódus bármi okból sikertelen, az ODBC.INI fájl változatlan marad, és hiba következik be.

Az új adatforrások hozzáadása a Windows Control Panel ODBC telepítőikonjával vagy a meglévő bejegyzések módosításával célszerű. Ha erre mégis a **RegisterDatabase** metódust használjuk, ajánlatos a *panel* argumentumot **True** értékre állítani.

Remove

Szintaxis 1: *objektum*.**Remove** {*index* | *könyvjelző*}

Szintaxis 2: *objektum*.**Remove** *index*

A **Remove** metódus két különböző szintaxisa eltérő elemeken érvényes, és eltérő funkciót lát el.

A Szintaxis 1. szerinti metódus **Columns** és **SelBookmarks** objektumgyűjteményen használható, és a **DBGrid** objektumbeli kijelölt sorok **SelBookmark** halmazából adott elemet vagy a **DBGrid** objektumbeli **Columns** halmazból egy **Columns** objektumot távolít el. A szintaxis argumentumai a következők:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum egy 0 és a **Count** tulajdonság értéke — 1 közötti egész szám lehet.
- A *könyvjelző* argumentum a **DBGrid** vezérlőelem kijelölt cellájában tárolt, formázatlan adatot tartalmazó karakteres kifejezés.

A **Remove** metódus megszünteti a **DBGrid** vezérlőelem egy sorának kiválasztását, eltávolítja a könyvjelzőt a **SelBookmarks** halmazból, és a vezérlőelem **Count** tulajdonságának értékét eggyel növeli. Az eltávolítandó sorokat a sorindex vagy a sor könyvjelzőjének alapján lehet kijelölni. A **SelBookmarks** halmazban nem található sor kijelölése kezelhető hibát okoz.

A Szintaxis 2. szerinti metódus **Collection** objektumon használható, és az objektum

A Szintaxis 2. szerinti metódus **Collection** objektumon használható, és az objektum kijelölt sorának eltávolítására használható. A szintaxis argumentumai a következők:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum az eltávolítandó elemnek a halmazban történő pozícióját megadó kifejezés. Ha ez numerikus kifejezés, értéke 1 és a **Count** tulajdonság értéke közötti egész szám lehet. Ha az *index* karakteres kifejezés, annak az eltávolítandó elem halmazhoz adásakor használt *key* argumentumnak megfelelőnek kell lennie. A halmazban nem található elem kijelölése hibát okoz.

RemoveItem

Szintaxis: *objektum*.**RemoveItem** *index*

A **ComboBox**, **Grid** és **ListBox** vezérlőelemen használható metódussal egy **ComboBox** vagy **ListBox** vezérlőelemből egy elem, a **Grid** vezérlőelemből pedig egy sor távolítható el. A szintaxis argumentumai a következők:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum az eltávolítandó elemet vagy sort az objektumon belül azonosító egész szám. A **ComboBox** vagy **ListBox** első elemét, illetve a **Grid** első sorát az *index* nulla értéke jelöli.

A **Data** vezérlőelemhez kötött **ComboBox** vagy **ListBox** nem támogatja a **RemoveItem** metódust.

RepairDatabase

Szintaxis: **DBEngine.RepairDatabase** *adatbázis*

A **DBEngine** objektumon használható metódus megkísérli az *adatbázis* argumentumban adott adatbázis sérülésének javítását. Az *adatbázis* egy karakteres kifejezés, amely egy létező Microsoft Jet adatbázisfájl nevét és elérési útját tartalmazza. Ha az elérési út hiányzik, a fájl keresése csak az aktuális könyvtárban történik. Az egységes névadási konvenciót támogató rendszerben hálózati elérési út is megadható, például a "\\MYSERVER\MYSHARE\MYDIR\MYDB.MDB" alakban.

Az *adatbázis* javításának megkezdése előtt azt be kell zárni, és a többfelhasználós környezetben más felhasználók sem tarthatják nyitva a javítás alatt. Ha az *adatbázis* nincs bezárva vagy nem kizárólagos használatra van megnyitva, hiba következik be.

A metódus a befejezetlen írási művelet következtében valószínűleg hibásként jelölt adatbázis javítását kísérli meg. Ez az eset akkor következhet be, ha egy, a Microsoft Jet adatbázis-szerkesztőt használó alkalmazás tápfeszültség kiesés vagy hardver okokból váratlanul bezáródik. A **Close** metódus használata vagy az alkalmazásból a szokásos módon történő kilépés után az adatbázis nem lehet valószínűleg sérültnek jelölve.

ségének ellenőrzését is. A nem javítható adatok törlődnek. Ha az adatbázis nem javítható, kezelhető hiba következik be.

Hibás adatbázis megnyitásának vagy tömörítésének kísérlete általában kezelhető hibát okoz, noha egyes esetekben az adatbázis sérülése nem érzékelhető és így erre hibajelzés sem figyelmeztethet. A felhasználói programban az adatbázis kiszámíthatatlan viselkedése esetére célszerű lehetőséget adni a felhasználó számára a **RepairDatabase** metódus használatára.

Egyes adatbázistípusok egy alkalmazásból a **Database** vagy **Recordset** objektumok vagy a Microsoft Jet bezárása nélkül történő kilépéskor sérülhetnek. Az ilyen sérülések elkerülésére célszerű olyan eljárásokat készíteni az alkalmazások bezárására és a rendszerből történő kilépésre, amelyek biztosítják az összes oldal adatbázisba mentését, illetve szünetmentes tápegységet kell használni az esetleges tápfeszültség ingadozások hatásának megelőzésére.

Az adatbázis helyreállítását követően hasznos lépés a **CompactDatabase** metódus használatával az adatbázisfájl töredezettségét megszüntetni, és ezzel a tárolásához szükséges lemezterületet csökkenteni.

Requery

Szintaxis: *rekordhalmaz*.**Requery** [*újQueryDef*]

Ez a eredményhalmaz és pillanatfelvétel típusú **Recordset** objektumon használható metódus az objektumban az objektum alapjául szolgáló lekérdezés újbóli végrehajtásával frissíti az adatokat. A metódus nem használható tábla típusú **Recordset** objektumon, továbbá az olyan eredményhalmaz és pillanatfelvétel típusú **Recordset** objektumon, amely **Restartable** tulajdonsága **False** értékű. A szintaxis argumentumai a következők:

- A *rekordhalmaz* argumentum egy **Recordset** adattípusúként deklarált változó.
- A feltételes *újQueryDef* argumentum egy **QueryDef** adattípusúként deklarált változó.

A **Requery** metódus használatával biztosítható, hogy a **Recordset** a legújabb adatokat tartalmazza. A metódus használatakor a többi felhasználó által a táblán végrehajtott változtatások megjelennek a **Recordset** objektumban, és annak első rekordja válik az aktuális rekorddá.

Amennyiben a metódus végrehajtásakor a **Recordset** objektum **BOF** és **EOF** tulajdonságai egyaránt **True** értékűek, a lekérdezés nem ad vissza rekordot, és a **Recordset** nem fog adatot tartalmazni.

A **Requery** metódus az aktuális lekérdezési paraméterekkel vagy az *újQueryDef* argumentumban adottakkal ismételten feltölti az aktuális **Recordset** objektumot.

Ha az eredeti **QueryDef** rendelkezésre áll, az eredetileg ezzel adott argumentumokat használó **Recordset** szükséges. Ha a **Recordset** készítése **QueryDef** nélkül történt, vagy itt nem a készítéséhez használt **QueryDef** van megadva, a **Recordset** lényegében

vagy itt nem a készítéséhez használt **QueryDef** van megadva, a **Recordset** lényegében rögtönzött módon jött létre. Megjegyzendő, hogy a **Restart** tulajdonság használatával állapítható meg, hogy ez a lekérdezés újrafuttatható, és a **Requery** a **CopyQueeryDef** metódussal összhangban használható-e.

Rollback

Szintaxis 1: *munkaterület*.**Rollback**

Szintaxis 2: *adatbázis*. **Rollback**

Ez a **Workspace** objektumon használható metódus az objektum által meghatározott munkamenet során lezár egy tranzakciót, és a **Workspace** objektumban lévő adatbázist visszaállítja a tranzakciót megkezdésekor érvényes állapotába. Noha a metódus a **Database** objektumon is használható, ez nem javasolt, mivel ezek a **Database** metódusok elavultak. A metódus szintaxisai a következő objektumazonosítókat tartalmazzák:

- A *munkaterület* azonosítója a **Workspace** objektumot megadó objektumtípusú változó.
- Az *adatbázis* azonosítója a **Database** objektumot megadó objektumtípusú változó.

A metódust a **Workspace** vagy **Database** objektumokkal akkor használhatjuk, ha az egyetlen munkamenetben végzett változtatások sorozatát vissza akarjuk vonni. A **Database** objektummal történő használata esetén a metódus a **Database** objektum **Workspace** objektumán alkalmazódik. A hasonló nevű tranzakciós utasítás használata nem javasolt.

Egy **Workspace** objektumon belül a tranzakciók globálisak a **Workspace** objektumra nézve, és nem csak egy adatbázisra vagy rekordhalmazra korlátozottak.

Ha az egynél több adatbázison vagy rekordkészleten hajtunk végre műveleteket, egy tranzakción belül a változásokat a **Rollback** metódussal lehet érvényteleníteni.

A **CommitTrans** használatát követően az adatbázison abban a munkamenetben végzett változtatások már nem vonhatók vissza, hacsak a tranzakció nem egy másik tranzakcióba van ágyazva, amelyben a **Rollback** metódussal a visszavonás még lehetséges. Egy **Workspace** objektumban, többszörösen egymásba ágyazott **BeginTrans** és **CommitTrans** vagy **Rollback** kombinációk használata esetén a tranzakciók öt szintje lehet egyszerre nyitva. Az egymásba ágyazott tranzakciók esetén egy szinttel csak a tranzakció lezárása és a változások mentése, illetve visszavonása után lehet visszalépni. Ha egyidejű, átlapolt és nem egymásba ágyazott hatókörű tranzakcióra van szükség, a versengő tranzakciókat tartalmazó járulékos **Workspace** objektumok hozhatók létre. A külső ODBC SQL adatbázisok nem teszik lehetővé egymásba ágyazott tranzakciók használatát.

A **Workspace** objektumnak a függőben lévő tranzakciók mentése vagy visszavonása

nélküli lezárása esetén a tranzakciók hatása automatikusan visszavonódik.

A **RollBack** metódus használatát megelőzően egy **BeginTrans** metódus kell használni, különben hiba következik be.

Egy tranzakció megkezdésekor a Microsoft Jet a műveleteit egy, a munkaállomás TEMP környezeti változójában megadott fájlba rögzíti. Ha a tranzakció naplózása során a TEMP meghajtóján rendelkezésre álló területet betelik, hiba (2004) következik be. Ha ilyenkor a **CommitTrans** metódus van használatban, meghatározhatatlan számú művelet végrehajtása történik meg, a fennmaradók azonban elvesznek, és a műveletet újból kell indítani. A tranzakciós naplózást a **Rollback** metódus használatával lehet felszabadítani, és a tranzakción belüli összes műveletet visszavonni.

RowBookmark

Szintaxis: *objektum*.**RowBookmark** *érték*

Ez a **DBGrid** vezérlőelemen használható metódus a vezérlőelemben látható sorhoz tartalmazó könyvjelzőt tartalmazó értéket ad vissza. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum egy 0 és a **DBGrid** vezérlőelem **VisibleRows** tulajdonságának értéke -1 közötti egész szám.

A **RowBookmark**(0) metódus ugyanazon könyvjelzőt adja vissza, mint a **DBGrid** vezérlőelem **FirstRow** tulajdonsága. A **DBGrid** vezérlőelem **Bookmark** tulajdonságával meghatározott, rejtett állapotú aktuális sor nem adható vissza ezzel a metódussal.

A metódus által visszaadott könyvjelzők nem menthetők, mivel a vezérlőelemben látható sorok változásakor értékeik azonnal megváltoznak.

RowContaining

Szintaxis: *objektum*.**RowContaining** *koordináta*

Ez a **DBGrid** vezérlőelemen használható metódus a vezérlőelem adott függőleges koordinátáját a sorok számának megfelelő értékként adja vissza. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *koordináta* argumentum egy numerikus kifejezés, amely a befogadó koordinátarendszeren alapuló függőleges koordinátát határozza meg. Értéke 0 és a **DBGrid** vezérlőelem **VisibleRows** tulajdonságának értéke -1 közötti egész szám lehet. A befogadó koordinátarendszerén kívülre eső *koordináta* kezelhető hibát okoz.

A **RowContaining** metódus az objektum által adott vezérlőelem egy oszlopindexének értékét adja vissza. A metódus különösen az egérrel történő munka és az elhúzási események esetén hasznos annak meghatározására, hogy a felhasználó kattintott vagy elhúzott-e egy másik, a **DBGrid** egy soraként megadott objektumot.

RowTop

Szintaxis: *objektum*.**RowTop** *érték*

Ez a **DBGrid** vezérlőelemen használható metódus a vezérlőelem adott sorának legfelső függőleges koordinátáját tartalmazó értéket ad vissza. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *koordináta* argumentum egy egész szám, amelynek értéke 0 és a **DBGrid** vezérlőelem **VisibleRows** tulajdonságának értéke -1 közötti egész szám lehet.

A metódus egy, az objektum *érték* által adott sorának tetejét jelölő és a befogadó **ScaleMode** tulajdonságán alapuló koordinátaértéket ad vissza.

A **RowTop** metódust a **Column** objektum **RowHeight**, **Left** és **Width** tulajdonságával együtt a **DBGrid** vezérlőelemben kiválasztott cella pontos helyének és méreteinek meghatározására lehet használni.

SaveToFile

Szintaxis: *objektum*. **SaveToFile** *fájlszám*

Ez a metódus **OLE** befogadó vezérlőelemen használható egy objektum adatfájlba mentésére. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *fájlszám* argumentum egy numerikus kifejezés, amelynek értéke az objektum mentésére használandó fájt jelöli ki. A fájlszámnak egy megnyitott, bináris fájlra kell mutatnia.

A metódust OLE objektumok mentésére használhatjuk, de az **OLE** formában történő mentésére a **SaveToOle1File** metódus alkalmas.

Ha az objektum csatolt, azaz az **OLEType** = **vbOLELinked** (vagy 0) beállítású, csak a csatolási információk és az adat egy képe kerül a fájlba. Az adatokat az objektumot létrehozó alkalmazás tartja karban. Amennyiben az objektum beágyazott, azaz az **OLEType** = **vbOLEEmbedded** (vagy 1) beállítású, az objektum adatait az OLE befogadó vezérlőelem tartja karban és a Visual Basic alkalmazásunk menti.

Az adatfájlba mentett objektum betöltésére a **ReadFromFile** metódus alkalmas.

SaveToOle1File

Szintaxis: *objektum*. **SaveToOle1File** *fájlszám*

Ez a metódus **OLE** befogadó vezérlőelemen használható egy objektumnak OLE formájú fájlba mentésére. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *fájlszám* argumentum egy numerikus kifejezés, amelynek értéke az objektum mentésére használandó fájt jelöli ki. A fájlszámnak egy megnyitott, bináris fájlra kell mutatnia

Ha az objektum csatolt, azaz az **OLEType** = **vbOLELinked** (vagy 0) beállítású, csak a csatolási információk és az adat egy képe kerül a fájlba. Az adatokat az objektumot létrehozó alkalmazás tartja karban. Amennyiben az objektum beágyazott, azaz az **OLEType** = **vbOLEEmbedded** (vagy 1) beállítású, az objektum adatait az OLE befogadó vezérlőelem tartja karban, és a Visual Basic alkalmazásunk menti.

Az OLE objektumok mentésére a **SaveToFile**, betöltésére pedig a **ReadFromFile** metódus alkalmas.

Scale

Szintaxis: [*objektum*].**Scale** [(*x1*, *y1*) - (*x2*, *y2*)]

A metódussal **Form** és **Printer** objektumon, valamint **PictureBox** vezérlőelemen lehet koordinátarendszert definiálni. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha nincs megadva, az *objektum* az éppen fókuszban lévő **Form** lesz.
- A feltételes *x1*, *y1* argumentum egy **Single** típusú értékpár, amely az objektum bal felső sarkának koordinátáit határozza meg. Az argumentumnak zárójelben kell szerepelnie. Csak az *x2*, *y2* argumentummal együtt használható.
- A feltételes *x2*, *y2* argumentum egy **Single** típusú értékpár, amely az objektum jobb alsó sarkának koordinátáit határozza meg. Az argumentumnak zárójelben kell szerepelnie. Csak az *x1*, *y1* argumentummal együtt használható.

A metódus a koordinátarendszer újbóli beállítására ad lehetőséget, és a futásidejű grafikai utasításokra és a vezérlőelemek helyzetére egyaránt hat.

A **Scale** argumentum nélküli használata a koordinátarendszert twip egységekre állítja vissza. Egy (logikai) cm = 567 twip.

ScaleX

Szintaxis: [*objektum*].**ScaleX** *szélesség* [*erről*, *erre*]

A **Form** és **Printer** objektumon, valamint **PictureBox** vezérlőelemen érvényes metódussal az érvényes szélesség mértékegysége állítható be. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha nincs megadva, az *objektum* az éppen fókuszban lévő **Form** lesz.
- A *szélesség* argumentum a módosítandó objektum mértékegységének számát adja meg.
- A feltételes *erről* argumentum egy szimbolikus konstans vagy **Integer** típusú érték, amely azt adja meg, hogy a módosítás miről történik. Az értékek a **ScaleMode** tulajdonság értékei és a **HiMetric** lehetnek.
- A feltételes *erre* argumentum egy szimbolikus konstans vagy **Integer** típusú érték, amely azt adja meg, hogy a módosítás mire történik. Az értékek a **ScaleMode** tulajdonság értékei és a **HiMetric** lehetnek.

Az argumentumban szereplő *erről* és *erre* a következők valamelyike lehet:

Ha a szimbolikus konstans:	Értéke:	A mértékegység:
vbUser	0	Felhasználó által definiált: az *objektum* szélességét a felhasználó állíthatja be.
vbTwips	1	Twip: 567 twip = 1 (logikai) cm, azaz 1440 twip = 1 (logikai) inch.
vbPoints	2	Pont: 72 pont = 1 (logikai) inch.
vbPixels	3	Képpont: a monitor vagy nyomtató legkisebb egysége.
vbCharacters	4	Karakter: vízszintesen 120 twip, függőlegesen 240 twip.
vbInches	5	Inch (kb. 25, 4 mm)
vbMillimeters	6	Milliméter
vbCentimeters	7	Centiméter
	8	HiMetric egység. Az *erről* argumentum hiánya esetén alapértelmezett.

A **ScaleX** metódus a szélesség *erről* argumentumban adott mértékegységét az *erre* argumentum által meghatározott egységre változtatja.

A **ScaleX** és **ScaleY** metódusok a **PaintPicture** metódussal együtt is használhatók.

ScaleY

Szintaxis: [*objektum*].**ScaleY** *magasság* [*erről*, *erre*]

A **Form** és **Printer** objektumon, valamint **PictureBox** vezérlőelemen érvényes metódussal az érvényes magasság mértékegysége állítható be. A metódusra vonatkozó tudnivalók értelemszerűen azonosak a **ScaleX** metódusnál leírtakkal.

Scroll

Szintaxis: *objektum*.**Scroll** *oszlopérték*, *sorérték*

A **DBGrid** vezérlőelemen érvényes metódus a vezérlőelemet egyetlen művelettel vízszintes és függőleges irányban gördíti. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *oszlopérték* argumentum egy **Long** típusú numerikus kifejezés, amely a vezérlőelemben egy oszlopot határoz meg.
- A *sorérték* argumentum egy **Long** típusú numerikus kifejezés, amely a vezérlőelemben egy sort határoz meg.

A pozitív *oszlopérték* jobbra, a *sorérték* pedig lefelé gördít. A tartományt túllépő értékek nem okoznak hibát, hatásukra a vezérlőelem az adott irányú szélső helyzetbe kerül. Ugyanez a **TopRow** és **Left** tulajdonságok beállításával is elérhető, de ezt csak két lépésben, a tulajdonságok egymástól független beállításával, és így két **Paint** eseményt okozva tehetjük meg.

Seek

Szintaxis: *rekordhalmaz*.**Seek** *összehasonlítás*, *kulcs1* ,*kulcs2* ...

Ez a csak indexelt tábla típusú **Recordset** (valamint az elavult **Table**) objektumon használható metódus meghatározza az objektumon belül egy adott feltételnek megfelelő rekord helyét, majd azt aktuális rekorddá teszi. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- A *rekordhalmaz* azonosítója egy létező, tábla típusú **Recordset** (valamint az elavult **Table**) objektum neve, amelynek az objektum **Index** tulajdonságában adottnak megfelelő indexe van.
- Az *összehasonlítás* argumentum a következő karakteres kifejezések egyike: "<", "<=", "=", ">=" vagy ">".
- A *kulcs1* ,*kulcs2* ... argumentumok az alapul szolgáló tábla **Primery** tulajdonság értékének megfelelő egy vagy több érték.

A metódus használata előtt az aktuális indexet az **Index** tulajdonság használatával be kell állítani. A metódus átvizsgálja az adott kulcsmezőket, meghatározza az első, az *összehasonlítás* és *kulcs1* feltételnek megfelelő rekord helyét, azt aktuális rekorddá teszi, majd a **NoMatch** tulajdonságot **False** értékre állítja. Amennyiben nem található a feltételeknek megfelelő rekord, a **NoMatch** tulajdonság **True** marad, az aktuális rekord pedig határozatlan lesz. A metódus sikerességét a **NoMatch** tulajdonság értékével mindig ellenőrizni kell.

Ha az *összehasonlítás* argumentum "=", ">=" vagy ">", a metódus az index elejétől, ha pedig "<" vagy "<=", az index végénél kezdi a keresést, kivéve ha többszörös indexbejegyzések találhatók a végén. Ebben az esetben a metódus a többszörös indexbejegyzések között egy tetszőleges bejegyzéstől kezdi a keresést. Ha az index több kulcsmezőt is kijelöl, a **Seek** metódus a feltételnek megfelelő elsőt keresi meg.

A *kulcs1* argumentumnak ugyanazon mezőadat típusúnak (például numerikusnak vagy szövegesnek) kell lennie, mint a megfelelő mezőnek az aktuális indexben.

A létező indexek számát az **Indexes** halmazzal lehet meghatározni. A **Seek** használatakor nem lehet beállított aktuális rekord.

A metódus semmilyen típusú hozzáfűzött táblán sem használható, mivel azt a **Seek** metódust nem támogató eredményhalmaz vagy pillanatfelvétel típusú **Recordset**, valamint **Dynaset** vagy **Snapshot** objektumként kell megnyitni, azonban az **OpenDatabase** metódussal közvetlenül megnyitott, telepíthető ISAM adatbázis tábláin már használható.

Adott feltételnek megfelelő rekordnak eredményhalmaz vagy pillanatfelvétel típusú **Recordset** objektumban történő kereséséhez valamelyik **Find** (**FindFirst**, **FindPrevious**, **FindNext** vagy **FindLast**) metódust kell használni. Az összes rekordon az egyik rekordról másikra feltétel nélküli történő mozgáshoz valamelyik **Move** metódus alkalmas.

Az aktuális rekord szerkesztésekor a következő rekordra lépés előtt az **Update** metódussal a változtatást érvényesíteni kell, különben azok figyelmeztetés nélkül elvesznek.

SetData

Szintaxis: *objektum*.**SetData** *adat* [, *forma*]

Ez a **Clipboard** objektumon érvényes metódus az adott grafikus formának megfelelő képet helyez a vágólapra. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- . Az *adat* argumentum a **Clipboard** objektumra helyezendő grafika.
- A feltételes *forma* argumentum egy szimbolikus konstans vagy **Integer** típusú érték, amely a Visual Basic által használt valamelyik grafikus formát adja meg. Ha az argumentum hiányzik, a **SetData** automatikusan határozza meg a grafikus formát.

Az argumentumban szereplő *forma* a következők valamelyike lehet:

Ha a szimbolikus konstans:	Értéke:	A grafikus forma:
dbCFBitmap	2	Bitkép (.BMP)
dbCFMetafile	3	Metafájl (.WMF)
dbCFDIB	8	Eszközfüggetlen bitkép (DIB)
dbCFPalette	9	Színpaletta

A **Clipboard** objektumra helyezendő képet vagy a **LoadPicture** függvénnyel, vagy egy **Form**, **Image** vagy **PictureBox Picture** tulajdonságával lehet kijelölni

SetFocus

Szintaxis: *objektum*.**SetFocus**

A metódus a fókuszt fogadni képes, látható állapotú **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Form**, **HScrollBar**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **TextBox** vagy **VScrollBar** vezérlőelemen, valamint **Form** vagy **MDIForm** űrlapon használható a fókusz beállítására. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **SetFocus** metódus használata után minden felhasználói beviteli művelet az objektumként adott vezérlőelemre vagy objektumra irányul.

Mivel az űrlap és az űrlapon lévő vezérlőelemek mindaddig nem láthatók, amíg az űrlap **Load** eseménye le nem zárul, a **SetFocus** metódus használatához a **Form_Load** eseményeljárás befejezése előtt a **Show** metódus használatával meg kell jeleníteni az űrlapot.

A rejtett vezérlőelemeken és űrlapokon kívül a metódussal a **False** értékű **Enabled** tulajdonságú űrlap vagy vezérlőelem sem tehető kiválaszthatóvá. Ha a **False** beállítás fejlesztésidőben történt, a **SetFocus** alkalmazása előtt elsőként az **Enabled** tulajdonságot **True** értékre kell változtatni.

SetText

Szintaxis: *objektum*.**SetText** *adat* [, *forma*]

A metódus **Clipboard** objektumon használható, és adott **Clipboard** objektumformájú szöveget helyez az objektumra. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *adat* argumentum a **Clipboard** objektumra helyezendő szöveg.
- A feltételes *forma* argumentum egy szimbolikus konstans vagy **Integer** típusú érték, amely a Visual Basic által használt valamelyik karakteres formát adja meg.

Az argumentumban szereplő *forma* a következők valamelyike lehet:

Ha a szimbolikus konstans:	Értéke:	A forma:
dbCFLink	&HBF00	DDE párbeszéd információ
dbCFRTF	&HBF01	Rich Text alak
dbCFText	1	Szöveg (Alapértelmezett)

Show

Szintaxis: [*objektum*].**Show** [*stílus*]

A metódussal **Form** és **MDIForm** objektum jeleníthető meg a képernyőn. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá.
- A feltételes *stílus* argumentum egy egész szám, amely az űrlap modalitását határozza meg: ha a *stílus* értéke 0, az űrlap modalitás nélküli, ha pedig 1, az űrlap modális lesz.

Ha a **Show** metódusban a memóriába még nem betöltött űrlapot adunk meg, a Visual Basic azt automatikusan betölti.

Amikor a metódus nem modális űrlapot jelenít meg, a metódust követő parancs végrehajtása az előfordulásakor azonnal megtörténik, míg modális űrlap esetében a parancs végrehajtása mindaddig nem lehetséges, amíg az űrlap rejtett vagy nincs betöltve. Az **MDIForm** nem lehet modális.

Modális űrlap megjelenítése alatt — a modális űrlapon lévő objektumokra irányulók kivételével — nem történhetnek billentyűzetről vagy egérről érkező események. A modális űrlapot a programnak rejtetté kell tenni vagy el kell távolítani, mielőtt más űrlapra irányuló beviteli művelet történhetne. Bár az alkalmazásban a modális űrlap megjelenítése alatt a többi űrlap letiltott, az egyéb alkalmazások nem.

Az alkalmazási program bejelentkező űrlapja a **Load** esemény bekövetkezéskor automatikusan, a **Show** metódus használata nélkül jelenik meg.

ShowColor

Szintaxis: *objektum*.**ShowColor**

A metódus **CommonDialog** vezérlőelemen használható, és annak Color párbeszédpaneljét jeleníti meg. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

ShowFont

Szintaxis: *objektum*.**ShowFont**

A metódus **CommonDialog** vezérlőelemen használható, és annak Font párbeszédpaneljét jeleníti meg. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A metódus használata előtt a **CommonDialog** vezérlőelem **Flags** tulajdonságát a következő értékek valamelyikére kell beállítani: **cdlCFScreenFonts** (&H1), **cdlCFPrinterFonts** (&H2) vagy **cdlCFBoth** (&H3). A beállítás hiánya futásidejű hibát okoz és a "There are no fonts installed" (Nincs telepített font) üzenetpanelt jeleníti meg.

ShowHelp

Szintaxis: *objektum*.**ShowHelp**

A metódust **CommonDialog** vezérlőelemen használhatjuk a WINHELP.EXE fájl futtatására és egy adott súgófájl megjelenítésére. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A metódus használata előtt a **CommonDialog** vezérlőelem **HelpFile** és **HelpCommand** tulajdonságát a megfelelő szimbolikus konstansra vagy értékre kell beállítani, különben a WINHELP.EXE nem jeleníti meg a súgó fájlt.

ShowOpen

Szintaxis: *objektum*.**ShowOpen**

A metódus **CommonDialog** vezérlőelemen használható, és annak Open párbeszédpaneljét jeleníti meg. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

ShowPrinter

Szintaxis: *objektum*.**ShowPrinter**

A metódus **CommonDialog** vezérlőelemen használható, és annak Printer párbeszédpaneljét jeleníti meg. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

ShowSave

Szintaxis: *objektum*.**ShowSave**

A metódus **CommonDialog** vezérlőelemen használható, és annak Save As párbeszédpaneljét jeleníti meg. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

ShowWhatsThis

Szintaxis: *objektum*.**ShowWhatsThis**

A metódust csak Windows 95 (vagy Windows NT 3.51) alatt lehet használni a **CheckBox**, **ComboBox**, **CommandButton**, **Data**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemeken a Windows 95 súgója által adott What's This előugró panelen a súgófájlban kiválasztott témakör megjelenítésére.

A **ShowWhatsThis** metódus az alkalmazás témaérzékeny súgójának készítéséhez alkalmas a szintaxisban adott *objektum* **ShowThisHelpID** tulajdonságával kijelölt témakör megjelenítésével.

Synchronize

Szintaxis: *adatbázis*.**Synchronize**(*elérési_út* [, *változási_irány*])

A metódus a **Database** aktuális, többszörözhető adatbázis objektumot előírt irányban szinkronizálja egy adott .MDB fájlban. A metódus szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *adatbázis* azonosítója a szinkronizálandó másolat neve.
- Az *elérési_út* argumentum annak a céladatbázisnak az elérési útja, amelyhez az *adatbázis* szinkronizálása történik. Az elérési út Unicode vagy hálózati könyvtárnevet is tartalmazhat. A fájlnévbe a kiterjesztés (.MDB) automatikusan beleértődik, így a kiterjesztés megadása nem kötelező.
- A feltételes *változási_irány* argumentum egy szimbolikus konstans, amely a két adatbázis közötti szinkronizálás irányát határozza meg.

A *változási_irány* argumentum lehetséges értékei a következők lehetnek:

Ha a szimbolikus konstans:	Hatása:
dbRepExportChanges	A változásokat az aktuális adatbázistól az *elérési_út* argumentumban adott adatbázis felé küldi.
dbRepImportChanges	A változásokat az aktuális adatbázis az *elérési_út* argumentumban adott adatbázistól kapja.

dbRepImpExpChanges Kétirányú adatcsere. (Alapértelmezett.)

A metódus az aktuális adatbázis és az *elérési_út* argumentumban adott .MDB adatbázis között biztosítja az adat- és sémaváltozások cseréjét. Az adatcsere első lépése mindig a sémaváltozások átvitele, mivel az adatcsere előtt mindkét adatbázisnak ugyanazon sémaszinten kell lennie.

Az *elérési_út* argumentumban adott adatbázis másolatnak ugyanazon másolathalmaz részének kell lennie. Ha mindkét másolat **ReplicaID** tulajdonsága vagy állapota azonos, a szinkronizálás sikertelen lesz.

TextHeight

Szintaxis: [*objektum*].**TextHeight**(*karakteres_kifejezés*)

A metódus **Form** és **Printer** objektumon, valamint **PictureBox** vezérlőelemen érvényes, és az objektumon az aktuális karakternagysággal megjelenő karakteres kifejezés megjelenítésének függőleges helyszükségletét adja vissza. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá
- A *karakteres_kifejezés* argumentum azt a karakteres kifejezést tartalmazza, amelynek magasságát meg kívánjuk határozni. Megadásához a zárójelek kötelezőek.

A magasság az adott objektumon érvényes **ScaleMode** tulajdonság beállításának vagy a **Scale** metódus koordinátarendszerének megfelelő egységben értendő. A magasság meghatározása a karakterek feletti és alatti helyigényt és a többsoros, azaz beágyazott kocsivisssza-soremelést is tartalmazó kifejezés esetén a sorok számát is figyelembe véve történik.

TextWidth

Szintaxis: [*objektum*].**TextWidth**(*karakteres_kifejezés*)

A metódus **Form** és **Printer** objektumon, valamint **PictureBox** vezérlőelemen érvényes, és az objektumon az aktuális karakternagysággal megjelenő karakteres kifejezés megjelenítésének vízszintes helyszükségletét adja vissza. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá
- A *karakteres_kifejezés* argumentum azt a karakteres kifejezést tartalmazza, amely szélességét meg kívánjuk határozni. Megadásához a zárójelek kötelezőek.

A szélesség az adott objektumon érvényes **ScaleMode** tulajdonság beállításának vagy a **Scale** metódus koordinátarendszerének megfelelő egységben értendő. A szélesség meghatározása a többsoros, azaz beágyazott kocsivisssza-soremelést is tartalmazó kifejezés esetén a legszélesebb sornak megfelelően történik.

Update

Szintaxis 1: *rekordhalmaz*.**Update**

Szintaxis 2: *objektum*.**Update**

Az **Update** metódust kétféle objektumcsoporton, eltérő szintaxissal és eltérő hatással használhatjuk.

A Szintaxis 1. szerinti metódus a **Dynaset**, eredményhalmaz és tábla típusú **Recordset** (valamint az elavult **Table**) objektumokon érvényes, és a másolatpuffer tartalmát a *rekordhalmaz* azonosítóban adott nevű objektumba menti, vagyis a metódust az aktuális rekord mentésére használhatjuk. A rekordon végzett módosítások elvesznek:

- Ha az **Edit** vagy **AddNew** metódust használtuk, és azután az **Update** használata nélkül másik rekordra léptünk át.
- Ha az **Edit** vagy **AddNew** metódust használtuk és azután az **Update** használata nélkül ismét egy **Edit** vagy **AddNew** metódust hívunk meg.
- Ha a **Bookmark** tulajdonságot másik rekordra állítjuk.
- Ha a *rekordhalmaz* azonosítóban adott nevű objektumot az **Update** használata nélkül zárjuk be.

Az **Update** használatát megkísérelve hibát kapunk, ha a szerkesztés megkezdése előtt az aktuális rekord tartalmát nem másoltuk a másolatpufferbe.

Amikor a többfelhasználós környezetben a **Recordset** objektum **LockEdit** tulajdonságának értéke **True**, vagyis pesszimisztikus zárolást használunk, a rekord az **Edit** használatától az **Update** metódus befejeződéséig zárolt marad. Ha a **LockEdit** tulajdonság értéke **False**, vagyis optimisztikus zárolást használunk, a rekord zárolt marad és az adatbázisbeli frissítése előtt a szerkesztés előtti állapottal hasonlítódik össze. Ha a rekord az **Edit** metódus használata óta változott, az **Update** metódus kezelhető hibával (3197) zárul. A külső (ODBC és telepíthető ISAM) adatbázisokon mindig az optimisztikus zárolás használatos. Az **Update** metódus ismételt használatával folytathatjuk a frissítési műveletet saját változtatásainkkal. A módosított rekordra a **Move** metódussal vagy a **Bookmark** tulajdonságát saját magára beállítva lehet visszatérni.

Egy cikluson belül bármely, egynél több rekordot szerkesztő DAO (Data Access Object) metódus használatakor (**AddNew**, **Delete**, **Edit** vagy **Update**) a Microsoft Jet 3.0-s verzió a teljes ciklust egy implicit tranzakcióként kezeli, vagyis minden egyes frissítést nem szükségszerűen írhatjuk azonnal az adatbázisba, ami hatást gyakorolhat a hibakezelésre, az oldalzárolásra és a változásoknak a többi felhasználó általi láthatóságára.

A változások azonnali adatbázisba írásához egy explicit tranzakción belül minden egyes **Edit...Update** blokkot be kell zárni.

Az **Update** metódus a Szintaxis 2 szerint **OLE** befogadó objektumokon érvényes, és az objektumot szolgáltató alkalmazásból az aktuális adatot grafikaként jeleníti meg az adott objektumban. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

UpdateControls

Szintaxis: *objektum*.**UpdateControls**

A metódus **Data** vezérlőelemen használható, és annak **Recordset** objektumából az aktuális rekordot a **Data** vezérlőelemhez kötött vezérlőelemekben jeleníti meg. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az **UpdateControls** a vezérlőelemek eredeti értékének visszaállítására használható, például ha a felhasználó valamilyen adatmódosítást végzett, és azt vissza kívánja vonni. A metódus hatása azonos az aktuális rekord ismét aktuálissá tételével, azonban nem jár semmilyen esemény bekövetkezésével.

UpdateRecord

Szintaxis: *objektum*.**UpdateRecord**

A metódus **Data** vezérlőelemen használható, és az ahhoz kötött vezérlőelemek aktuális értékét menti. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az **UpdateRecord** használatakor a kötött vezérlőelemek aktuális értékeinek az érvényes adatbázisba mentése a **Validate** eseményt bekövetkezése nélkül történik, megakadályozva ezzel a kaszkád, egymás után következő események létrejöttét.

A metódus hatása azonos egy **Edit** metódus végrehajtását követően egy mező tartalmának módosításával és annak **Update** metódussal történő érvényesítésével, azzal az eltéréssel, hogy az **UpdateRecord** metódus használata nem jár semmilyen esemény bekövetkezésével, vagyis így a **Validate** esemény elkerülhetővé válik.

Egy rekordnak az adatbázisban történő frissítését megkísérelve, a rekordnak minden érvényességi feltételnek meg kell felelnie az adatbázisba írásához. Ezeket a feltételeket a **ValidationRule** tulajdonság — a Microsoft SQL Server esetén a Transact SQL alapértelmezések, szabályok és triggerek — beállítása határozza meg.

Az olyan esetekben, ha művelet sértené a hivatkozási integritási korlátokat, a rekordot tartalmazó lap zárolt, az adatbázis vagy **Recordset** objektum nem frissíthető, vagy a felhasználónak nincs jogosultsága a művelet végrehajtására, a frissítés nem történik meg és kezelhető hiba következik be.

WhatsThisMode

Szintaxis: *objektum*.**WhatsThisMode**

A csak Windows 95 (vagy Windows NT 3.51) alatt a **Form** vagy **MDIForm** objektumon használható metódus hatására az egérmutató alakja a kérdőjel mutatóra változik és az alkalmazást előkészíti a kijelölt objektumra vonatkozó a What's This súgópanel megjelenítésére. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **WhatsThisMode** metódus végrehajtása az alkalmazást ugyanazon állapotba viszi, mintha a címsorban a What's This gombra kattintottunk volna. Ha a felhasználó egy objektumra kattint, ezzel az adott objektum **WhatsThisHelpID** tulajdonságának megfelelő témaérzékeny súgóját hívja meg. A metódus különösen a súgónak az alkalmazás menüsorának egy menüjéből történő használatára alkalmas.

ZOrder

Szintaxis: [*objektum*].**ZOrder** [*pozíció*]

A metódus **App**, **Form**, **MDIForm** és **Screen** objektumokon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **Line**, **ListBox**, **OptionButton**, **PictureBox**, **Shape**, **TextBox** és **VScrollBar** vezérlőelemeken használható az adott objektum vagy vezérlőelem grafikai fedést meghatározó sorban elfoglalt helyének legelőre vagy leghátulra helyezésére. A metódus szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- A feltételes *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá
- A feltételes *pozíció* argumentum egy egész szám, amely az *objektum* pozícióját ugyanezen *objektum* egy másik példányához viszonyítottan határozza meg. Ha a *pozíció* értéke 0 vagy hiányzik, az *objektum* a legelső, ha pedig 1, a legutolsó lesz a sorban.

Az objektumok sorbeli helyzete a programfejlesztési fázisban az Edit menü Bring To Front (legelső pozíció) vagy Send To Back (leghátsó pozíció) parancsával állítható be.

A *pozíció* értékétől függően a **ZOrder** metódus az **MDIForm** objektumokban az MDI-alablakot az MDI-kliensterülete elé vagy mögé helyezi, a **Form** és **MDIForm** objektumokon pedig a képernyőn a legelső (látható) vagy a leghátsó (más futó alkalmazásokkal fedett) helyzetbe hozza.

Mint tudjuk, az űrlapokkal és vezérlőelemekkel három grafikai réteg van kapcsolatban: a leghátsón jelennek meg a grafikus metódusok eredményei, a középső a grafikus objektumokat és **Label** vezérlőelemeket tartalmazza, míg a legelső a **CommandButton**,

CheckBox, **ListBox** és hasonló, nem grafikus vezérlőelemek jelennek meg. A réteg tartalma takarja a mögötte lévő réteg(ek) tartalmát. A **ZOrder** az objektumokat csak azon a rétegen belül rendezi át, ahol azok megjelenhetnek.

Események

Activate

Szintaxis: **Private Sub** *objektum*_**Activate**( )

Ez az esemény **Form** és **MDIForm** objektumon következik be, ha az adott objektum (egy aktív állapotú űrlap vagy MDI-ablak) aktívvá válik. Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az űrlap akkor lesz aktív állapotú, ha a felhasználó rákattint, illetve ha programkódban a **Show** vagy **SetFocus** metódus aktívvá teszi. Az esemény feltétele, hogy az adott űrlap látható legyen, vagyis egy űrlapot a **Load** utasítással történő betöltés után még a **Show** metódussal vagy a **Visible** tulajdonságának **True** beállításával láthatóvá kell tenni.

Az **Activate** esemény csak az azonos alkalmazási programon belül lévő elemek kiválaszthatóságának, fókuszba kerülésének változásakor következhet be, másik alkalmazásra történő átlépés nem okozza az esemény bekövetkezését.

Az **MDIForm** objektumban esetén csak az egyes alablakok közötti fókuszváltás vált ki **Activate** eseményt, amelyet mindig a megfelelő alablak kap meg, azonban ha a fókuszváltás egy **MDIForm** alablaka és egy **Form** alablaka közötti következik be, az **Activate** eseményt az **MDIForm** kapja. Az **Activate** megelőzi a **GetFocus** eseményt.

AfterColUpdate

Szintaxis: **Private Sub** *objektum*_**AfterColUpdate**([*index* **As Integer** ,] *oszlopindex* **As Integer**)

Ez az esemény a **DBGrid** vezérlőelemen azután következik be, hogy a vezérlőelem egy celljából adat kerül a másolatpufferbe. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- Az *oszlopindex* argumentum a vezérlőelembeli oszlopot megadó **Integer** típusú szám.

Amikor a felhasználó egy **DBGrid** vezérlőelem-cellában a szövegbeviteli műveletet

akár ugyanazon sor másik oszlopába átlépve, akár az ENTER billentyűt leütve befejezi, vagy amikor a vezérlőelem elveszti a fókuszt, megtörténik a **BeforeColUpdate** eseményhez tartozó eljárás végrehajtása, majd — hacsak annak *letiltás* argumentuma nem **True** értékű — az adat a cellából a másolatpufferbe kerül, és egy **AfterColUpdate** esemény következik be.

Amint egy **AfterColUpdate** eseményeljárás végrehajtása elkezdődött, a cella adata már nem érvényteleníthető (hiszen az már átkerült a vezérlőelem másolatpufferébe), azonban a **Recordset** rekordhalmazba helyezése előtt más frissítési művelettel módosítható.

AfterDelete

Szintaxis: **Private Sub** *objektum*_**AfterDelete(** [*index* **As Integer** ,] *oszlopindex* **As Integer**)

Ez az esemény a **DBGrid** vezérlőelemen egy kijelölt rekord törlése után következik be. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- Az *oszlopindex* argumentum a vezérlőelembeli oszlopot megadó **Integer** típusú szám.

Amikor a felhasználó egy **DBGrid** vezérlőelemben kiválasztott egy rekordot, és leüti a `DEL` vagy `CTRL+X` billentyűket, a kijelölt sor törlődik. A törlést megelőzően egy **BeforeDelete**, majd a törlését követően egy **AfterDelete** esemény következik be. A törlésre kijelölt sor a **SelBookmarks** által adott rekordhalmazként érhető el.

AfterInsert

Szintaxis: **Private Sub** *objektum*_**AfterInsert(** *index* **As Integer)**

Ez az esemény a **DBGrid** vezérlőelemen azután következik be, hogy a vezérlőelembe egy új rekordot illesztünk. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

Amikor a felhasználó egy új rekordot jelöl ki a vezérlőelem alján, és a cellák egyikébe egy karakter jegyez be, egy **BeforeInsert**, **BeforeUpdate**, **AfterUpdate** és végül

egy **AfterInsert** esemény következik be.

Az **AfterInsert** esemény bekövetkeztekor a rekord már hozzáadódott az adatbázishoz, és a **Bookmark** tulajdonságon keresztül lehet hozzáférni. Az **AfterInsert** esemény nem érvényteleníthető.

Az eseményt más táblák frissítéséhez vagy más vezérlőelemek frissítés utáni törlésének végrehajtására lehet használni.

AfterUpdate

Szintaxis: **Private Sub** *objektum*_**AfterUpdate(** *index* **As Integer)**

Ez az esemény a **DBGrid** vezérlőelemen azután következik be, hogy a vezérlőelemről a módosított adat az adatbázisba íródott. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

Amikor a felhasználó egy másik sorra lép vagy a **Recordset** objektum **Update** metódusát használja, az adat a vezérlőelem másolatpufferéből a **Data** vezérlőelem pufferébe kerül, majd az adatbázisba íródik. A beírás befejeződése egy **AfterUpdate** eseményt okoz.

A frissített rekordhoz a **DBGrid** vezérlőelem **Bookmark** tulajdonságon keresztül lehet hozzáférni.

Az **AfterUpdate** esemény a **BeforeUpdate** után, de a vezérlőelem **LostFocus** eseménye vagy a léptetési sor következő vezérlőelemének **GotFocus** eseménye előtt következik be, még akkor is, ha a vezérlőelem nem kötött. Az esemény nem érvényteleníthető, azonban más frissítési művelettel módosítható.

Eltérően a **Change** eseménytől, a vezérlőelemen vagy rekordban programkóddal történő adatmódosítás nem okoz **AfterUpdate** eseményt.

BeforeColUpdate

Szintaxis: **Private Sub** *objektum*_**BeforeColUpdate(** [*index* **As Integer** ,] *oszlopindex* **As Integer**, *régi_érték* **As Variant** [, *letiltás* **As Integer**])

Ez az esemény a **DBGrid** vezérlőelemen azután következik be, hogy a vezérlőelem egy cellájában a szerkesztés befejeződött, de az adat még került a másolatpufferbe. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

- Az *oszlopindex* argumentum a vezérlőelembeli oszlopot megadó **Integer** típusú szám.
- A *régi_érték* argumentum egy **Variant** adat, amely a cella módosítás előtti értékét tartalmazza.
- A feltételes *letiltás* argumentum egy **Boolean** típusú kifejezés, amely azt adja meg, hogy a változtatás megtörténjen-e.

A *letiltás* argumentum lehetséges értékei a következő hatásúak:

True Törli a módosítást, a cella tartalmát a *régi_érték* argumentumnak megfelelőre állítja vissza és a fókusz a vezérlőelemen marad.
False Folytatja a módosítást és engedélyezi a fókusz változását. (Alapértelmezett.)

A *régi_érték* argumentumban adott adat a vezérlőelem cellájából akkor kerül a másolatpufferbe, ha a felhasználó a szövegbeviteli műveletet akár ugyanazon sor másik oszlopába átlépve, akár az ENTER billentyűt leütve befejezi, vagy a fókuszt a celláról máshová viszi át. Az adat átvitele előtt egy **BeforeColUpdate** esemény következik be, amely lehetőséget ad a felhasználó számára a cella tartalmának ellenőrzésére.

Ha a *letiltás* argumentum értéke **True**, a cella tartalma az előző értékre áll vissza, a fókuszt pedig mindaddig nem lehet a vezérlőelemről elvinni, amíg az alkalmazás meg nem határozza, hogy az adat biztonságosan visszavihető-e a vezérlőelem másolatpufferébe. Természetesen ilyenkor nem következik be az **AfterColUpdate** esemény sem. A *régi_érték* cellába visszaállítása és a fókusznak a celláról történő elvitelének engedélyezése a **False** értékű *letiltás* argumentummal történhet.

BeforeDelete

Szintaxis: **Private Sub** *objektum*_**BeforeDelete(** [*index* **As Integer** [, *letiltás* **As Integer**]])

Ez az esemény a **DBGrid** vezérlőelemen azelőtt következik be, hogy a kijelölt rekord a vezérlőelemben törlődne. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A feltételes *letiltás* argumentum egy **Boolean** típusú kifejezés, amely azt adja meg, hogy a rekord törlése megtörténjen-e.

A *letiltás* argumentum lehetséges értékei a következő hatásúak:

True Nem törli a rekordot és a fókusz a vezérlőelemen marad.
False Folytatja a törlést és engedélyezi a fókusz változását. (Alapértelmezett.)

Amikor a felhasználó egy **DBGrid** vezérlőelemben kiválasztott egy rekordot, és leüti a `DEL` vagy `CTRL+X` billentyűket, a kijelölt sor törlését megelőzően egy **BeforeDelete**, majd a törlését követően egy **AfterDelete** esemény következik be. A törlésre kijelölt sor a **SelBookmarks** által adott rekordhalmazként érhető el. Több sor kijelölése esetén a "`Multiple rows cannot be deleted`" hibaüzenetet jelenik meg.

BeforeInsert

Szintaxis: **Private Sub** *objektum*_**BeforeInsert(** [*index* **As Integer** [, *letiltás* **As Integer**]])

Ez az esemény a **DBGrid** vezérlőelemen azelőtt következik be, hogy a vezérlőelembe egy új rekordot illesztünk. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A feltételes *letiltás* argumentum egy **Boolean** típusú kifejezés, amely azt adja meg, hogy a rekord hozzáadása megtörténjen-e.

A *letiltás* argumentum lehetséges értékei a következő hatásúak:

True Nem adja hozzá az új rekordot, a cellát törli és a fókusz a vezérlőelemen marad.
False Folytatja a hozzáadást és engedélyezi a fókusz változását. (Alapértelmezett.)

Amikor a felhasználó egy új rekordot jelöl ki a vezérlőelem alján, és a cellák egyikébe egy karakter jegyez be, egy **BeforeInsert**, **BeforeUpdate**, **AfterUpdate** és végül egy **AfterInsert** esemény következik be.

A **BeforeInsert** esemény bekövetkeztekor a rekord még nem adódott az adatbázishoz, hanem az eseményeljárás befejezéséig a **DBGrid** vezérlőelem másolatpufferében marad. Az **AfterInsert** esemény befejeződését követően az új rekord sora a **DBGrid** vezérlőelemben újrainicializálódik, és a szerkesztett rekord a vezérlőelem utolsó sorává válik.

BeforeUpdate

Szintaxis: **Private Sub** *objektum*_**BeforeUpdate(** [*index* **As Integer** [, *letiltás* **As Integer**]])

Ez az esemény a **DBGrid** vezérlőelemen azelőtt következik be, hogy a vezérlőelemből az adat a másolatpufferbe íródott volna. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A feltételes *letiltás* argumentum egy **Boolean** típusú kifejezés, amely azt adja meg, hogy a rekord másolása megtörténjen-e.

Amikor a felhasználó egy másik sorra lép, vagy a **Recordset** objektum **Update** metódusát használja, az adat a vezérlőelem másolatpufferéből a **Data** vezérlőelem pufferébe kerül, majd onnan az adatbázisba íródik. A beírás befejeződése egy **AfterUpdate** eseményt okoz.

Mielőtt az adat a **DBGrid** vezérlőelem másolatpufferéből a **Data** vezérlőelem másolatpufferébe kerülne, egy **BeforeUpdate** esemény, majd az adatnak a másolatpufferbe visszakerülése és az adatbázisba írása után — hacsak a másolás nincs a **True** értékű *letiltás* argumentummal letiltva — egy **AfterUpdate** esemény következik be.

A frissített rekordhoz a **DBGrid** vezérlőelem **Bookmark** tulajdonságon keresztül lehet hozzáférni.

Ha a *letiltás* argumentum értéke **True**, a fókusz a vezérlőelemen marad, sem **AfterUpdate**, sem **LostFocus** esemény nem következik be, valamint az adat sem kerül az adatbázisba.

A **BeforeUpdate** esemény még akkor is a vezérlőelem **AfterUpdate** és **LostFocus** eseménye vagy a léptetési sor következő vezérlőelemének **GotFocus** eseménye előtt következik be, ha a vezérlőelem nem kötött.

Eltérően a **Change** eseménytől, a vezérlőelemen vagy rekordban programkóddal történő adatmódosítás nem okoz **BeforeUpdate** eseményt.

Change

Szintaxis: **Private Sub** *objektum*_**Change(** [*index* **As Integer**] **)**

A **Change** esemény a következő vezérlőelemek tartalmának változása esetén következik be: **ComboBox**, **DBCombo**, **DBGrid**, **DirListBox**, **DriveListBox**, **HScrollBar**, **Label**, **PictureBox**, **TextBox** és **VScrollBar**. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

Az egyes vezérlőelemtől függően az esemény bekövetkezésének hatása és okozója a következő:

ComboBox A vezérlőelem szövegét módosítja a szövegbeviteli mezőben. Az ese-

mény a **Style** tulajdonság 0 (*Dropdown Combo*) vagy 1 (*Simple Combo*) értékei esetén a felhasználó által végrehajtott szövegmódosításkor vagy a **Text** tulajdonság értékének programkódból történő módosításakor következik be.

DirListBox Módosítja a kijelölt könyvtárt. Az esemény új könyvtárnév dupla kattintással történő kijelölésekor vagy a **Path** tulajdonság értékének programkódból történő módosításakor következik be.

DriveListBox Módosítja a kijelölt meghajtót. Az esemény új meghajtó kijelölésekor vagy a **Drive** tulajdonság értékének programkódból történő módosításakor következik be.

HScrollBar Elmozgatja a vízszintes gördítősáv csúszkáját. Az esemény a csúszka elmozdításakor vagy a **Value** tulajdonság értékének programkódból történő módosításakor következik be.

Label Módosítja a **Label** tartalmát. Az esemény egy DDE kapcsolat által végrehajtott adatfrissítéskor vagy a **Caption** tulajdonság értékének programkódból történő módosításakor következik be.

PictureBox Módosítja a **PictureBox** tartalmát. Az esemény egy DDE kapcsolat által végrehajtott adatfrissítéskor vagy a **Picture** tulajdonság értékének programkódból történő módosításakor következik be.

TextBox Módosítja a **TextBox** tartalmát. Az esemény egy DDE kapcsolat által végrehajtott adatfrissítéskor, a felhasználó által végzett szövegmódosításkor vagy a **Text** tulajdonság értékének programkódból történő módosításakor következik be.

VScrollBar Elmozgatja a függőleges gördítősáv csúszkáját. Az esemény a csúszka elmozdításakor vagy a **Value** tulajdonság értékének programkódból történő módosításakor következik be.

A **Change** eseményeljárás a vezérlőelemek közötti koordinátaadatok kijelzésére, az adatok szinkronizálására, összhangba hozására vagy a fájlrendszerrel kapcsolatos vezérlőelemek tulajdonságainak frissítésére is alkalmas.

Az eljárás bizonyos körülmények között kaszkád események bekövetkezését okozhatja, például ha egy vezérlőelem **Change** eseménye ugyanazon vezérlőelem tartalmát módosítja. Az ilyen eseménysor elkerüléséhez a következő szempontokat kell figyelembe venni:

- Ha lehetséges, a vezérlőelemeken kerülni kell az olyan **Change** eseményeljárás használatát, amely módosítja a vezérlőelem tartalmát. Ha mégis ilyet kell használni, mindaddig le kell tiltani a további változásokat, amíg az eljárás feldolgozása be nem fejeződött.
- El kell kerülni két vagy több olyan vezérlőelem használatát, amelyek **Change** eseményeljárásai egymásra hatnak.
- El kell kerülni egy **MsgBox** függvény vagy utasítás használatát a **HScrollBar** és

VScrollBar vezérlőelemek **Change** eseményeljárásában.

Click

Szintaxis 1: **Private Sub Form_Click( )**

Szintaxis 2: **Private Sub** *objektum*_**Click(** [*index* **As Integer**] **)**

Ez az esemény a Szintaxis 1 szerint **Form** objektumon vagy a Szintaxis 2 szerint **Menu** objektumokon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **FileListBox**, **Frame**, **Grid**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox** és **TextBox** vezérlőelemeken akkor következhet be, ha a felhasználó valamelyik objektum felett lenyomja, majd felengedi az egérgombot, vagy amikor egy vezérlőelem értéke megváltozik. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

Az esemény a **Form** objektum esetén egy üres területére vagy egy letiltott vezérlőelemére történő kattintáskor következik be, míg a vezérlőelemeken a következő felhasználói műveletek eredményeznek **Click** eseményt:

- Egy vezérlőelemre kattintás a `JOBB` vagy `BAL` gombbal. A **CheckBox**, **CommandButton** vagy **OptionButton** vezérlőelemeken csak `BAL` egérgombbal történő kattintás érvényes.
- Egy elem kiválasztása a **ComboBox** vagy **ListBox** vezérlőelemen a nyíl lenyomásával vagy az elemre kattintással.
- A fókuszban lévő **CheckBox**, **CommandButton** vagy **OptionButton** vezérlőelem esetén a `SZÓKÖZ` billentyű leütésével.
- Űrlapon lévő **CommandButton** vezérlőelem **Default** tulajdonságának **True** értéke esetén az `ENTER` billentyű leütésével.
- Űrlapon lévő **Cancel** gomb, azaz egy **CommandButton** vezérlőelem **Cancel** tulajdonságának **True** értéke esetén az `ESC` billentyű leütésével.
- Hívóbetűs vezérlőelem esetén a hívóbetű leütésével.

Click eseményt a programkódból is előidézhetünk:

- A **CommandButton** vezérlőelem **Value** tulajdonságát **True** értékre állítva.
- Az **OptionButton** vezérlőelem **Value** tulajdonságát **True** értékre állítva.
- A **CheckBox** vezérlőelem **Value** tulajdonságának megváltoztatásával.

A vezérlőelemek állapota azok **Value** tulajdonságának programból történő kiolvasásával vizsgálható.

A **Click** eseményeljárást **CommandButton** vagy **PictureBox** vezérlőelemeken, illetve **Menu** objektumon parancsok vagy parancsszerű műveletek végrehajtásához használhatjuk. A vezérlőelemeken az egérgomb lenyomása és felengedése a vezérlőelemtől függő sorrendben **MouseDown**, **MouseUp** és **Click** eseményeket inicializál, például a **CommandButton** és **ListBox** esetén a sorrend **MouseDown** — **Click** — **MouseUp**. Ha az alkalmazás szempontjából fontos ezen események sorrendje, azt előzetesen ellenőrizni kell.

A JOBB, BAL és KÖZÉPSŐ egérgombok megkülönböztetését igénylő eljárásokhoz a **MouseDown** és **MouseUp** események használhatók.

ColResize

Szintaxis: **Private Sub** *objektum*_**ColResize(** [*index* **As Integer** ,] *oszlopindex* **As Integer** [, *letiltás* **As Integer**])

Ez az esemény **DBGrid** vezérlőelemen egy **Paint** eseményt megelőzően következik be, amikor a vezérlőelem oszlopait újraméretezi. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- Az *oszlopindex* argumentum a vezérlőelembeli oszlopot megadó **Integer** típusú szám.
- A feltételes *letiltás* argumentum egy **Boolean** típusú kifejezés, amely azt adja meg, hogy a változtatás megtörténjen-e.

A *letiltás* argumentum lehetséges értékei a következő hatásúak:

True Törli a módosítást, az oszlopot az eredeti nagyságra állítja vissza és törli a folyamatban lévő **Paint** eseményt.

False Folytatja a szélesség módosítását és a **Paint** eseményt. (Alapértelmezett.)

Amikor a felhasználó átméretez egy oszlopot, **ColResize** esemény következik be, amely elfogadhatja, részben fogadhatja el vagy törölheti a változtatást. A változtatás részbeni elfogadása a **Column.Width** tulajdonság megfelelő értékre beállításával történhet.

Az eljáráson belül végrehajtott **Refresh** metódus még akkor is a vezérlőelem frissítését eredményezi, ha a *letiltás* argumentum **True**.

DblClick

Szintaxis 1: **Private Sub Form_DblClick( )**

Szintaxis 2: **Private Sub** *objektum*_**DblClick(** *index* **As Integer)**

Ez a **Click** eseményhez hasonló esemény a Szintaxis 1 szerint **Form** objektumon vagy a Szintaxis 2 szerint **ComboBox**, **DBCombo**, **DBGrid**, **DBList**, **FileListBox**, **Frame**, **Grid**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox** és **TextBox** vezérlőelemeken akkor következhet be, ha a felhasználó valamelyik objektumra duplán kattint, azaz lenyomja, majd felenged egy egérgombot és ezt megismétli. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

A **DblClick** esemény a **Form** objektum esetén egy üres területére vagy egy letiltott vezérlőelemére történő dupla kattintáskor következik be, míg a vezérlőelemeken a következő felhasználói műveletek eredményeznek **DblClick** eseményt:

- Egy vezérlőelemre történő dupla kattintás a bal gombbal.
- Egy **ComboBox** egy elemén történő dupla kattintás, ha a vezérlőelem **Style** tulajdonsága 1 (*Simple*) vagy egy **DBCombo**, **DBList**, **FileListBox** vagy **ListBox** vezérlőelemben van.

A **DblClick** eseményeljárást olyan összekapcsolt, egymás utáni műveletek automatikus végrehajtására használhatjuk, mint pl. egy ikonra kattintva egy ablak vagy egy dokumentum kinyitása, vagy egy lista elemének kiválasztása és a párbeszédpanel bezárása.

A Visual Basicben úgy érhetünk el gyorsbillentyűnek megfelelő hatást, ha a **DblClick** eseményeljárást egy lista vagy fájllista alapértelmezett, azaz olyan **CommandButton** gombjával használjuk tandem módon, amely **Default** tulajdonsága **True**. Ilyenkor az alapértelmezett gomb **Click** eseményét a **DblClick** eseményeljárás részeként lehet meghívni.

Az egéreseményt fogadó objektumok esetén az események sorrendje a következő: **MouseDown**, **MouseUp**, **Click**, **DblClick** és **MouseUp**.

Két egymás utáni kattintást a rendszer csak akkor tekint kettős kattintásnak, ha azok meghatározott időhatáron belül érkeznek, egyébként csak **Click** eseményként kezeli azokat. A kattintások közötti megengedett időtkülönbséget a Windows Control Panel használatával lehet beállítani.

Azok a vezérlőelemek, amelyek nem tudják fogadni a **DblClick** eseményt, a dupla kattintást két **Click** eseményként érzékelik.

A JOBB, BAL és KÖZÉPSŐ egérgombok megkülönböztetését igénylő eljárásokhoz a **MouseDown** és **MouseUp** események használhatók.

Deactivate

Szintaxis: **Private Sub** *objektum*_**Dectivate( )**

Ez az esemény **Form** és **MDIForm** objektumon akkor következik be, ha az adott objektumról (egy aktív állapotú űrlapról vagy MDI-ablakról) valamely felhasználói művelettel, például egy ablakon vagy vezérlőelemen történő kattintással másik ablakra váltunk, és a továbbiakban az válik aktívvá. Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Deactivate** esemény csak az azonos alkalmazási programon belül lévő elemek fókuszba kerülésének változásakor következhet be, másik alkalmazásra történő átlépés nem okozza ezt az eseményt.

Az **MDIForm** objektumban esetén csak az egyes alablakok közötti fókuszváltás vált ki **Deactivate** eseményt, amelyet a megfelelő alablak kap meg, azonban ha a fókuszváltás egy **MDIForm** alablaka és egy **Form** alablaka közötti következik be, a **Deactivate** eseményt az **MDIForm** kapja.

A **GetFocus** megelőzi a **Deactivate** eseményt.

DragDrop

Szintaxis 1: **Private Sub Form _DragDrop(***forrás* **As Control**, *x* **As Single**, *y* **As Single**)

Szintaxis 2: **Private Sub MDIForm_DragDrop(***forrás* **As Control**, *x* **As Single**, *y* **As Single**)

Szintaxis 3: **Private Sub** *objektum*_**DragDrop(** [*index* **As Integer**,] *forrás* **As Control**, *x* **As Single**, *y* **As Single**)

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint a **MDIForm** objektumon, a Szintaxis 3 szerint pedig **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemeken akkor következik be, ha a felhasználó lenyomott egérgombbal egy objektumot "megfog", az ablak vagy egy célobjektum felett "elhúzza" (*drag*), majd a gomb felengedésével "leejti" (*drop*), illetve ha a **Drag** metódus *művelet* argumentumának értéke 2. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A *forrás* az elhúzandó vezérlőelemet a tulajdonságokat vagy metódusokat is magában foglaló alakban határozza meg, például `Source.Visible = False`.

- Az *x* és *y* argumentumok a célűrlapon vagy vezérlőelemen belüli egérmutató pozíciójának x és y koordinátáit adják meg a **ScaleHeight**, **ScaleWidth**, **ScaleLeft** és **ScaleTop** tulajdonságok beállításának megfelelően.

A **DragDrop** eseményeljárást annak vezérlésére használhatjuk, hogy egy elhúzási művelet után mi történjen, például az elhúzott vezérlőelem az új helyen jelenjen meg.

Amennyiben a *forrás* argumentum többszörös vezérlőelemeket tartalmazhat:

- A megfelelő vezérlőelem típusát a **TypeOf** kulcsszót **If** utasítással együtt használva adjuk meg.
- A kívánt vezérlőelemet a vezérlőelem **Tag** tulajdonságával jelöljük ki, majd használjunk egy **DragDrop** eseményeljárást.

Az elhúzás elindítási módja a **DragMode** tulajdonság és a **Drag** metódus használatával adható meg. Amint az elhúzás megkezdődött, a **DragDrop** eseményt követő események a **DragDrop** eseményeljárásban kezelhetők.

DragOver

Szintaxis 1: **Private Sub Form_DragOver(** *forrás* **As Control**, *x* **As Single**, *y* **As Single**, *állapot* **As Integer**)

Szintaxis 2: **Private Sub MDIForm_DragOver(** *forrás* **As Control**, *x* **As Single**, *y* **As Single**, *állapot* **As Integer**)

Szintaxis 3: **Private Sub** *objektum*_**DragOver(** [*index* **As Integer**,] *forrás* **As Control**, *x* **As Single**, *y* **As Single**, *állapot* **As Integer**)

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint a **MDIForm** objektumon, a Szintaxis 3 szerint pedig **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemeken akkor következik be, ha a felhasználó lenyomott egérgombbal egy objektumot "megfog", és az űrlap vagy egy célobjektum felett éppen "elhúz" (*drag*). Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A *forrás* az elhúzandó vezérlőelemet a tulajdonságokat vagy metódusokat is magában foglaló alakban határozhatja meg, például `Source.Visible =False`.
- Az *x* és *y* argumentumok a célűrlapon vagy vezérlőelemen belüli egérmutató pozíciójának x és y koordinátáit adják meg a **ScaleHeight**, **ScaleWidth**, **ScaleLeft**

és **ScaleTop** tulajdonságok beállításának megfelelően.

- Az *állapot* argumentum egy egész szám, amely megfelel az elhúzás alatt álló vezérlőelem célűrlaphoz vagy vezérlőelemhez viszonyított pillanatnyi állapotának.

Az *állapot* argumentum lehetséges értékei és azok értelmezése a következő:

Ha az *állapot*:	A pillanatnyi állapot:
0	Belépés — a forrásobjektum a célobjektum tartományába lépett.
1	Kilépés — a forrásobjektum elhagyta a célobjektum tartományát.
2	Takarás — a forrásobjektum a célobjektum tartományán belül mozog.

A **DragOver** eseményt annak jelzésére használhatjuk, hogy egy elhúzási művelet egy érvényes célhoz viszonyítva hogyan mozog, az eseményeljárásában pedig előírhatjuk, hogy mi történjen az elhúzási művelet elindítása és befejezése között, például az érvényes céltartomány kiemelése a **BackColor** vagy **ForeColor** tulajdonság kóddal történt beállításával, vagy speciális elhúzási mutató megjelenítése a **DragIcon** vagy **MousePointer** tulajdonság kóddal történt beállításával.

Az *állapot* argumentum figyelésével állapothoz kötött műveleteket lehet végezni, például a 0 értékhez érvényes céltartomány kiemelését vagy speciális elhúzási mutató megjelenítését, az 1 értékhez pedig az eredeti megjelenés vagy az eredeti mutató visszaállítását kapcsolva. Ha egy vezérlőelem akkor kapja meg a **DragOver** eseményt, amikor az *állapot* értéke 0:

- A forrás vezérlőelemet a célobjektumba ejtve, az objektum egy **DragDrop** eseményt kap.
- Ha a forrás vezérlőelem nincs a célobjektumba ejtve, az objektum az *állapot* 1 értékre változásakor egy **DragOver** eseményt kap.

Az elhúzás elindítási módja a **DragMode** tulajdonság és a **Drag** metódus használatával adható meg.

DropDown

Szintaxis: **Private Sub** *objektum*_**DropDown(** [*index* **As Integer**] **)**

Ez az esemény egy **ComboBox** vezérlőelemen akkor következik be, amikor annak listarésze éppen kinyílik. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

A **DropDown** eseményeljárással a **ComboBox** vezérlőelemen történő kiválasztási művelet előtt a lista elemei frissíthetők, az **AddItem** és **RemoveItem** metódusok hasz-

nálatával pedig a listához új elemek adhatók és elemek távolíthatók el belőle.

Az esemény a **Style** = 1 (*Simple Combo*) beállításban nem engedélyezett.

Error

Szintaxis: **Private Sub** *objektum*_**Error** ([*index* **As Integer**,] *hibaszám* **As Integer**, *válasz* **As Integer**)

Ez a **Data** vezérlőelemen bekövetkező eseményt a nem Visual Basic program végrehajtásakor bekövetkező adathozzáférési hiba okozza. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A *hibaszám* argumentum egy hibaszám.
- A *válasz* argumentum a hibára adandó választ kijelölő szimbolikus konstans vagy érték.

A *válasz* lehetséges értékei a következők:

Ha a szimbolikus konstans:	Értéke:	A bekövetkező válasz:
vbDataErrContinue	0	Folytatás
vbDataErrDisplay	1	Hibaüzenet megjelenítése. (Alapértelmezett)

Az **Error** eseményeljárás használata akkor hasznos, ha a futásidejű hiba nem saját programunkon belül következik be — amikor azt a már ismert módon lekezelhetnénk — hanem a következő műveletek valamelyike okoz **Error** eseményt:

- A felhasználó egy **Data** vezérlőelem gombjára kattint.
- A **Data** vezérlőelem automatikusan megnyit egy adatbázist, és a **Form_Load** esemény után egy **Recordset** objektumot tölt be.
- Egy felhasználói vezérlőelem **MoveNext**, **AddNew** vagy **Delete** metódust hajt végre.

Ha az **Error** eseményhez nem adunk eseményeljárás-kódot, a Visual Basic a hibához tartozó hibaüzenetet jeleníti meg.

GotFocus

Szintaxis 1: **Private Sub Form_GotFocus( )**

Szintaxis 2: **Private Sub** *objektum*_**GotFocus(** [*index* **As Integer**] **)**

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint pedig

CheckBox, **ComboBox**, **CommandButton**, **DBCombo**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Grid**, **HScrollBar**, **ListBox**, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemeken akkor következik be, ha azok akár az egérrel rajtuk történő kattintással, a TAB billentyűvel rálépve, vagy a programkódban a **SetFocus** metódus használatával megkapják a fókuszt. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

A **GotFocus** eseményeljárást általában az olyan műveletekhez használjuk, amelyeket az adott űrlap vagy vezérlőelem első fókuszba kerülésekor akarunk végrehajtatni, így például a vezérlőelem funkciójára és használatára vonatkozó információk megjelenítését.

Egy vezérlőelem csak akkor kerülhet fókuszba, ha az **Enabled** és **Visible** tulajdonsága **True**, míg a **Form** objektum esetén a rajta lévő, látható vezérlőelemek egyikének **Enabled** tulajdonsága sem lehet **True**.

HeadClick

Szintaxis: **Private Sub** *objektum*_**HeadClick(** [*index* **As Integer**,] *oszlopindex* **As Integer**)

Ez az esemény a **DBGrid** vezérlőelemen következik be egy adott oszlop fejlécére kattintáskor. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- Az *oszlopindex* argumentum a vezérlőelembeli oszlopot megadó **Integer** típusú szám.

A **HeadClick** eseményeljárás például a felhasználó általi kattintással kijelölt oszlopon alapuló **Data** vezérlőelem rekordforrásának rendezéséhez használható.

KeyDown

Szintaxis 1: **Private Sub Form_KeyDown(** *billentyűkód* **As Integer**, *eltolás* **As Integer**)

Szintaxis 2: **Private Sub** *objektum*_**KeyDown(** [*index* **As Integer**,] *billentyűkód* **As Integer**, *eltolás* **As Integer**)

Ez az esemény akkor következik be, ha egy billentyű leütésekor a Szintaxis 1 szerint

egy **Form** objektum, a Szintaxis 2 szerint pedig egy **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Grid**, **HScrollBar**, **ListBox**, **OptionButton**, **PictureBox**, **TextBox** vagy **VScrollBar** vezérlőelem a fókuszban van. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A *billentyűkód* argumentum itt — eltérően a **KeyPress** eseménytől — nem a szabványos ANSI kódot jelent, hanem például **vbKeyF1** (F1 billentyű) vagy **vbKeyHome** (HOME billentyű) alakú szimbolikus konstanst, amelyhez tartozó billentyűt a Visual Basic objektumtárában az Object Browser használatával kereshetjük meg.
- Az *eltolás* argumentum a SHIFT, a CTRL és az ALT billentyűk állapotát jelölő bitmező. A bitmező legkevésbé értékes bitjei a SHIFT billentyűnél a 0. bit, a CTRL billentyűnél 1. bit és az ALT billentyűnél a 2. bit, értéke pedig ennek megfelelő sorrendben 1, 2 és 4 lehet. A mezőben tetszőleges számú (akár az összes bit is) előfordulhat, jelezve a billentyűk lehetséges lenyomási kombinációit, például az *eltolás* = 6 érték az ALT +CTRL billentyűket jelöli.

Az *eltolás* argumentum vizsgálatához eljárásszinten vagy egy modul deklarációs részében szimbolikus konstansokat lehet deklarálni, amelyek ezután bitmaszkként működnek, és bármilyen billentyűleütési kombináció vizsgálatára alkalmazhatók. Az *eltolás* egyes értékeihez tartozó szimbolikus konstansok a következők lehetnek:

Ha az *eltolás*:	A szimbolikus konstans:	Értelmezése:
1	**vbShiftMask**	A SHIFT billentyű bitmaszkja.
2	**vbCtrlMask**	A CTRL billentyű bitmaszkja.
4	**vbAltMask**	Az ALT billentyű bitmaszkja.

A billentyűleütést a fókuszban lévő űrlap vagy vezérlőobjektum fogadja. Egy vezérlőelem csak akkor kerülhet fókuszba, ha az **Enabled** és **Visible** tulajdonsága **True**, míg a **Form** objektum esetén a rajta lévő, látható vezérlőelemek egyikének **Enabled** tulajdonsága sem lehet **True**.

A **KeyDown** esemény általában a speciális funkció- és mutatóvezérlő- billentyűkhöz, a billentyűkombinációkhoz és standard billentyűzetmódosítókhoz, valamint a numerikus tasztatúra és a standard billentyűzet megkülönböztetéséhez használatos. Megjegyzendő, hogy nem mindegyik billentyű leütése vált ki **KeyDown** eseményt. A kivételek:

- Az ENTER, ha az űrlapon olyan **CommandButton** gomb van, amelynek **Default** tulajdonsága **True**.

- Az ESC, ha az űrlapon olyan **CommandButton** gomb van, amelynek **Cancel** tulajdonsága **True**.
- A TAB billentyű.

A **KeyDown** esemény — ellentétben a **KeyPress** eseménnyel — ezen három billentyű kivételével az összes billentyű lenyomását képes érzékelni, azonban a kis- és nagybetűket csak két argumentum (*billentyűkód* és *eltolás*) beállításával tudja megkülönböztetni.

Az *eltolás* globális állandóinak eljárásszintű vagy egy modul deklarációs részében történő elhelyezése, valamint a leütött billentyű bitmaszkkal való összehasonlítása az előző változatokból már ismert módon történhet.

Annak kihasználásával, hogy a **True** értékű **KeyPreview** tulajdonsággal rendelkező űrlap a **KeyDow** és **KeyUp** eseményeket a rajta lévő vezérlőelemeknél előbb kapja meg, globális billentyűzetkezelő rutinokat lehet készíteni.

KeyPress

Szintaxis 1: **Private Sub Form_KeyPress(** *ANSI_kód* As Integer)

Szintaxis 2: **Private Sub** *objektum*_**KeyPress(** [*index* **As Integer**,] *ANSI_kód* **As Integer**)

Ez az esemény a Szintaxis 1 szerint egy **Form** objektumon, a Szintaxis 2 szerint pedig egy **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Grid**, **HScrollBar**, **ListBox**, **OptionButton**, **PictureBox**, **TextBox** vagy **VScrollBar** vezérlőelemen akkor következik be, amikor a felhasználó egy ANSI billentyűt leüt és felenged. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- Az *ANSI_kód* argumentum itt — eltérően a **KeyDown** és **KeyUp** eseménytől — a szabványos numerikus ANSI kódot jelent. Az argumentum hivatkozással adódik át, megváltoztatása új karakter megjelenítését eredményezi. Ha értéke 0, az objektum nem kapja meg a billentyűleütést.

A billentyűleütést a fókuszban lévő űrlap vagy vezérlőobjektum fogadja. A **Form** objektum esetén az esemény fogadásának feltétele a rajta lévő látható vezérlőelemek mindegyikének **False** értékű **Enabled**, vagy **True** értékű **KeyPreview** tulajdonsága.

Az *ANSI_kód* argumentumként — eltérően a **KeyDown** és **KeyUp** eseményektől — szabványos ANSI kódot kell használni. Az esemény minden nyomtatható karaktert, a CTRL billentyűvel kombinálható szabványos betű- vagy speciális karaktert, továbbá az

ENTER vagy BACKSPACE billentyűket érzékeli. Mivel az eljárás az ANSI kódot használja, képes a karakterek kis- és nagybetűs alakjait is megkülönbözteti, viszont a **KeyPress** által nem ismert billentyűk (funkció-, szerkesztő- és mutatómozgató-billentyűk) leütését csak a **KeyDown** és **KeyUp** eseményen keresztül lehet kezelni. Az *ANSI_kód* értékének karakterré konvertálásához a **Chr( )** függvény, a karakter ANSI kóddá visszaalakításához pedig az **Asc( )** függvény használható.

A **KeyPress** eseményeljárás a **TextBox** vagy **ComboBox** vezérlőelemen keresztül történt billentyűleütések elfogására használható, így a lehetővé téve azok formázását és érvényességük azonnali ellenőrzését is.

Mivel a CTRL+@ ANSI kódja 0, és így az *ANSI_kód* argumentumban törli a billentyűleütést, azaz az objektum nem kap karaktert, ezen billentyűkombináció használatát kerülni kell az alkalmazásokban.

KeyUp

Szintaxis 1: **Private Sub Form_KeyUp(** *billentyűkód* **As Integer**, *eltolás* **As Integer)**

Szintaxis 2: **Private Sub** *objektum*_**KeyUp(** [*index* **As Integer**,] *billentyűkód* **As Integer**, *eltolás* **As Integer)**

Ez az esemény akkor következik be, ha egy billentyű felengedésekor a Szintaxis 1 szerint egy **Form** objektum, a Szintaxis 2 szerint pedig egy **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Grid**, **HScrollBar**, **ListBox**, **OptionButton**, **PictureBox**, **TextBox** vagy **VScrollBar** vezérlőelem a fókuszban van. Az eseményre vonatkozó tudnivalók értelemszerűen azonosak a **KeyDown** eljárásnál leírtakkal.

LinkClose

Szintaxis 1: **Private Sub Form_LinkClose( )**

Szintaxis 2: **Private Sub MDIForm_ LinkClose( )**

Szintaxis 3: **Private Sub** *objektum*_ **LinkClose(** [*index* **As Integer**] **)**

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint **MDIForm** objektumon, a Szintaxis 3 szerint pedig **Label**, **PictureBox** és **TextBox** vezérlőelemeken akkor következik be, ha a DDE-n keresztül folyó adatcserét bármely alkalmazás bármikor befejezi. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

Az eseményeljárás jellemző alkalmazási területe a DDE kapcsolat befejeződésről a felhasználó értesítése, valamint számára a hibakereséshez és a kapcsolat újbóli kiépítéséhez szükséges információk küldése. Rövid üzenetek küldéséhez inkább a **MsgBox** függvény használható.

LinkError

Szintaxis 1: **Private Sub Form_LinkError(** *hibaszám*)

Szintaxis 2: **Private Sub MDIForm_LinkError(** *hibaszám*)

Szintaxis 3: **Private Sub** *objektum*_**LinkClose(** [*index* **As Integer**,] *hibaszám* **As Integer)**

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint **MDIForm** objektumon, a Szintaxis 3 szerint pedig **Label**, **PictureBox** és **TextBox** vezérlőelemeken akkor következik be, ha a DDE-n keresztül folyó adatcsere során hiba következett be. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A *hibaszám* argumentum egy DDE hibaszám.

LinkError eseményt csak a nem Visual Basic program végrehajtásakor bekövetkező DDE hiba okoz. A hiba azonosítása a *hibaszám* értékének átadásával történik:

hibaszám:	A hiba oka:
1	A másik felhasználó rossz formában kért adatot. A hiba egymás után többször is előfordulhat, ahogy a Visual Basic megkísérli a másik alkalmazás által felismerhető formát megtalálni.
6	A célalkalmazás azután kísérelte meg a DDE párbeszéd folytatását, hogy a forrásűrlapon a **LinkMode** tulajdonság 0 (*None*) lett.
7	Már minden forráskapcsolat (forrásonként 128) használatban van.
8	A célvezérléseknél: egy automatikus kapcsolat vagy **LinkRequest** metódus sikertelenül kísérelte meg a vezérlőelemet és adatát aktualizálni. A forrásűrlapoknál: a célalkalmazás sikertelenül kísérelt meg adatot elhelyezni egy vezérlőelemben.
11	Nincs elegendő memória a DDE végrehajtásához.

A **LinkError** eseményeljárást a felhasználónak a bekövetkezett hibákról történő értesítéséhez lehet használni. Az eljárásba hibajavító programkódot vagy a kapcsolat újbóli létrehozásához szükséges hibakeresési információkat is be lehet építeni. Rövid üzenetek küldéséhez inkább a **MsgBox** függvény használható.

LinkExecute

Szintaxis: **Private Sub** *objektum*_**LinkExecute(** *parancs* **As String**, *elfogadás* **As Integer)**

Ez az esemény **Form** és **MDIForm** objektumokon akkor következik be, ha a DDE párbeszéd során a célalkalmazás a forrás által végrehajtandó parancsot küld. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *parancs* argumentum a célalkalmazás által küldendő, külön szintaxissal nem rendelkező, az alkalmazástól függő hatású karakteres parancskifejezés.
- Az *elfogadás* argumentum egy egész szám, amely azt közli a célalkalmazással, hogy a parancs végrehajtása megtörtént-e. A 0 érték a parancs végrehajtását, minden más érték a parancs visszautasítását jelenti. Az alapértelmezett érték -1 (visszautasítás).

Ha előzőleg nem volt **LinkError** eseményeljárás létrehozva, a Visual Basic visszautasítja a célalkalmazástól érkező parancskifejezést.

LinkNotify

Szintaxis: **Private Sub** *objektum*_**LinkNotify(** [*index* **As Integer**])

Ez az esemény **Label**, **PictureBox** és **TextBox** vezérlőelemeken akkor következik be, ha a forrás megváltoztatott egy DDE kapcsolattal meghatározott adatot, és a cél vezérlőelem **LinkMode** tulajdonsága 3 (*Notify*) értékű volt. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

A **LinkNotify** eseményeljárásban a felhasználót figyelmeztetni lehet az adatváltozásról, az új adat pedig azonnal vagy később áll rendelkezésre. Az új adathoz a **LinkRequest** metódus használatával juthatunk hozzá.

LinkOpen

Szintaxis 1: **Private Sub Form_LinkOpen(** *elfogadás* **As Integer**)

Szintaxis 2: **Private Sub MDIForm_LinkOpen(** *elfogadás* **As Integer**)

Szintaxis 3: **Private Sub** *objektum*_**LinkOpen(** [*index* **As Integer**,] *elfogadás* **As Integer**)

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint **MDIForm** objektumon akkor következik be, ha az űrlappal a célalkalmazás DDE párbeszédet kezdeményez, a Szintaxis 3 szerint pedig akkor, ha a forrásalkalmazással egy **PictureBox** vagy **TextBox** vezérlőelem kezdeményez DDE párbeszédet. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- Az *elfogadás* argumentum egy egész szám, amely azt közli a célalkalmazással, hogy a DDE párbeszéd létrehozása megtörtént-e. A 0 érték a létrehozása végrehajtását, minden más érték pedig a visszautasítását jelenti. Az alapértelmezett érték -1 (visszautasítás).

Load

Szintaxis 1: **Private Sub Form_Load( )**

Szintaxis 2: **Private Sub MDIForm_Load( )**

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint **MDIForm** objektumon egy űrlap betöltődésekor következik be. Az automatikus indítás űrlapja esetén az esemény a **Load** utasítással történt alkalmazásindítás vagy egy betöltetlen űrlap tulajdonságára vagy vezérlőelemére történt hivatkozás eredményeként következik be. Ez esetben az űrlap ugyan automatikusan betöltődik, de csak akkor lesz automatikusan látható állapotú is, ha **MDIChild** tulajdonsága **True** értékre volt beállítva. Amennyiben egy **MDIForm** nincs betöltve egy MDI-alablak betöltésekor, az **MDIForm** és az MDI-alablak is automatikusan betöltődik, és mindkettő látható állapotú lesz. Más űrlapokat betöltés után csak a **Show** metódussal vagy **Visible** tulajdonságuk **True** értékre állításával tehetjük láthatóvá.

A **Load** eseményeljárás jellemzően egy űrlap inicializálási, például a vezérlőelemek alapértelmezett értékeit, a **ComboBox** vagy **ListBox** vezérlőelemekbe töltendő adatokat, és az űrlapszintű változókat beállító kódját tartalmazhatja.

A betöltéssel kapcsolatos egyéb, például **Activate**, **GotFocus**, **Paint** vagy **Resize** események használatakor gondoskodni kell arról, hogy azok ne kerüljenek egymással ütközésbe, és ne okozzanak rekurzív eseményeket.

A **Load** esemény az **Initialize** esemény után következik.

LostFocus

Szintaxis 1: **Private Sub Form_LostFocus( )**

Szintaxis 2: **Private Sub** *objektum*_**LostFocus(** [*index* **As Integer**] **)**

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint pedig **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Grid**, **HScrollBar**, **ListBox**, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemeken akkor következik be, ha azok akár az egérrel más objektumra történt kattintással, akár a TAB billentyűvel más objektumra átlépve, vagy a programkódban a **SetFocus** metódus használatával a fókuszt megváltoztatva elveszítik a fókuszt. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

A **LostFocus** eseményeljárás az objektum elhagyásakor, például a frissítések ellenőrzésére és érvényesítésére, más objektumok engedélyezésére, letiltására, elrejtésére vagy megjelenítésére, továbbá a **GotFocus** eseményeljárásban beállított feltételek módosítására használható.

MouseDown

Szintaxis 1: **Private Sub Form_MouseDown(** *gomb* **As Integer**, *eltolás* **As Integer**, *y* **As Single**, *y* **As Single**)

Szintaxis 2: **Private Sub MDIForm_MouseDown(** *gomb* **As Integer**, *eltolás* **As Integer**, *y* **As Single**, *y* **As Single**)

Szintaxis 3: **Private Sub** *objektum*_**MouseDown(** [*index* **As Integer**,] *gomb* **As Integer**, *eltolás* **As Integer**, *y* **As Single**, *y* **As Single**)

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint a **MDIForm** objektumon, a Szintaxis 3 szerint pedig **CheckBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **FileListBox**, **Frame**, **Grid**, **Image**, **Label**, **ListBox**, **OptionButton**, **PictureBox** és **TextBox** vezérlőelemeken akkor következik be, ha a felhasználó lenyom egy egérgombot. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A *gomb* argumentum az eseményt okozó egérgombot azonosító bitmező, amelyben a 0. bit a BAL, az 1. bit a JOBB és a 2. bit a KÖZÉPSŐ gombot jelzi, ennek megfelelően értéke azonos sorrendben 1, 2 vagy 4 lehet. Csak egyetlen ilyen bit lehet beállított, mivel a **MouseMove** eseménybeli értelmezéstől eltérően a *gomb* itt (és a **MouseUp** eseménynél) csak egyetlen egérgomb lenyomását jelentheti.

- Az *eltolás* argumentum a SHIFT, a CTRL és az ALT billentyűk állapotát jelölő bitmező. A bitmező legkevésbé értékes bitjei a SHIFT billentyűnél a 0. bit, a CTRL billentyűnél 1. bit és az ALT billentyűnél a 2. bit, értéke pedig ennek megfelelő sorrendben 1, 2 és 4 lehet. A mezőben tetszőleges számú (akár az összes bit is) előfordulhat, jelezve a billentyűk lehetséges lenyomási kombinációit, például az *eltolás* = 6 érték az ALT +CTRL billentyűket jelöli.
- Az *x* és *y* argumentumok a célűrlapon vagy vezérlőelemen belüli egérmutató pozíciójának x és y koordinátáit adja meg a **ScaleHeight**, **ScaleWidth**, **ScaleLeft** és **ScaleTop** tulajdonságok beállításának megfelelően.

A *gomb* és *eltolás* argumentumok vizsgálatához eljárásszinten vagy egy modul deklarációs részében szimbolikus konstansokat lehet deklarálni, amelyek ezután bitmaszkként működnek, és bármilyen egérgomb- és billentyűlenyomási kombináció vizsgálatára alkalmazhatók.

A *gomb* egyes értékeihez tartozó szimbolikus konstansok a következők lehetnek:

Ha a *gomb*:	A szimbolikus konstans:	Értelmezése:
1	**vbLeftButton**	A JOBB gomb bitmaszkja.
2	**vbRightButton**	A BAL gomb bitmaszkja.
4	**vbMiddleButton**	A KÖZÉPSŐ gomb bitmaszkja.

Az *eltolás* egyes értékeihez tartozó szimbolikus konstansok a következők lehetnek:

Ha az *eltolás*:	A szimbolikus konstans:	Értelmezése:
1	**vbShiftMask**	A SHIFT billentyű bitmaszkja.
2	**vbCtrlMask**	A CTRL billentyű bitmaszkja.
4	**vbAltMask**	Az ALT billentyű bitmaszkja.

A **MouseDown** eseményeljárás az egérgombok lenyomásával kapcsolatos műveletek elvégzésére szolgáló utasításokhoz használatos. Eltérően a **Click** és **DblClick** eseményektől, a **MouseDown** képes megkülönböztetni az egyes egérgombokat, továbbá egy gomb lenyomását a SHIFT, CTRL és ALT billentyűkkel kombinálva is képes értelmezni.

A **MouseDown** és a **MouseUp** események között az egérrel kapcsolatos eseményeket az az objektum vagy űrlap kapja meg, amely felett a **MouseDown** esemény bekövetkeztekor az egérmutató éppen volt, még akkor is, ha az egérmutató pozíciója (az *x* és *y* koordináta) közben kikerül az adott objektum belsejéből.

Az egymást követő egérgombnyomásoknál az eseményeket egészen az utolsó gomb felengedéséig az első gombnyomást vevő objektum kapja.

MouseMove

Szintaxis 1: **Private Sub Form_MouseMove(** *gomb* **As Integer**, *eltolás* **As Integer**, *y* **As Single**, *y* **As Single)**

Szintaxis 2: **Private Sub MDIForm_MouseMove(** *gomb* **As Integer**, *eltolás* **As Integer**, *y* **As Single**, *y* **As Single)**

Szintaxis 3: **Private Sub** *objektum*_**MouseMove(** [*index* **As Integer**,] *gomb* **As Integer**, *eltolás* **As Integer**, *y* **As Single**, *y* **As Single)**

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint **MDIForm** objektumon, a Szintaxis 3 szerint pedig **CheckBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **FileListBox**, **Frame**, **Grid**, **Image**, **Label**, **ListBox**, **OptionButton**, **PictureBox** és **TextBox** vezérlőelemeken akkor következik be, ha a felhasználó mozgatja az egeret. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A *gomb* argumentum az eseményt okozó egérgombot azonosító bitmező, amelyben a 0. bit a `BAL`, az 1. bit a `JOBB` és a 2. bit a `KÖZÉPSŐ` gombot jelzi, ennek megfelelően értéke azonos sorrendben 1, 2 vagy 4 lehet. A mezőben tetszőleges számú (akár az összes bit is) előfordulhat, mivel a **MouseDown** és a **MouseUp** eseménybeli értelmezéstől eltérően a *gomb* itt egy, két vagy három gomb lenyomott állapotát is jelölheti.
- Az *eltolás* argumentum a `SHIFT`, a `CTRL` és az `ALT` billentyűk állapotát jelölő bitmező. A bitmező legkevésbé értékes bitjei a `SHIFT` billentyűnél a 0. bit, a `CTRL` billentyűnél 1. bit és az `ALT` billentyűnél a 2. bit, értéke pedig ennek megfelelő sorrendben 1, 2 és 4 lehet. A mezőben tetszőleges számú (akár az összes bit is) előfordulhat, jelezve a billentyűk lehetséges lenyomási kombinációit, például az *eltolás* = 6 érték az `ALT +CTRL` billentyűket jelöli.
- Az *x* és *y* argumentumok a célűrlapon vagy vezérlőelemen belüli egérmutató pozíciójának x és y koordinátáit adja meg a **ScaleHeight**, **ScaleWidth**, **ScaleLeft** és **ScaleTop** tulajdonságok beállításának megfelelően.

A *gomb* és *eltolás* argumentumok vizsgálatához eljárásszinten vagy egy modul deklarációs részében szimbolikus konstansokat lehet deklarálni, amelyek ezután bitmaszkként működnek, és bármilyen egérgomb- és billentyűlenyomási kombináció vizsgálatára alkalmazhatók.

A *gomb* egyes értékeihez tartozó szimbolikus konstansok a következők lehetnek:

Ha a *gomb*:	A szimbolikus konstans:	Értelmezése:
1	**vbLeftButton**	A JOBB gomb bitmaszkja.
2	**vbRightButton**	A BAL gomb bitmaszkja.
4	**vbMiddleButton**	A KÖZÉPSŐ gomb bitmaszkja.

Az *eltolás* egyes értékeihez tartozó szimbolikus konstansok a következők lehetnek:

Ha az *eltolás*:	A szimbolikus konstans:	Értelmezése:
1	**vbShiftMask**	A SHIFT billentyű bitmaszkja.
2	**vbCtrlMask**	A CTRL billentyű bitmaszkja.
4	**vbAltMask**	Az ALT billentyű bitmaszkja.

A **MouseMove** esemény az egérnek az objektumokon keresztüli mozgatása során folyamatosan generálódik, az egérműveleteket pedig az az objektum vagy űrlap kapja meg, amely határain belül az egér éppen mozog.

A **MouseMove** eseményeljárás általában az egér mozgatásával kapcsolatos műveletek elvégzésére szolgál. Eltérően a **MouseDown** és **MouseUp** eljárásoktól, a **MouseMove** képes mindhárom egérgomb állapotát meghatározni, míg az előbbiek csak egy adott gomb állapotát jelzik.

Ha a **MouseMove** eseményeljárásban egy űrlapot mozgatunk, ezzel kaszkád eseménysort okozunk, azaz álló egér mellett is bekövetkezhet **MouseMove** esemény, amelyet az egérmutató alatt mozgó űrlap okoz.

MouseUp

Szintaxis 1: **Private Sub Form_MouseUp(** *gomb* **As Integer**, *eltolás* **As Integer**, *y* **As Single**, *y* **As Single**)

Szintaxis 2: **Private Sub MDIForm_MouseUp(** *gomb* **As Integer**, *eltolás* **As Integer**, *y* **As Single**, *y* **As Single**)

Szintaxis 3: **Private Sub** *objektum*_**MouseUp(** [*index* **As Integer**,] *gomb* **As Integer**, *eltolás* **As Integer**, *y* **As Single**, *y* **As Single**)

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint **MDIForm** objektumon, a Szintaxis 3 szerint pedig **CheckBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **FileListBox**, **Frame**, **Grid**, **Image**, **Label**, **ListBox**, **OptionButton**, **PictureBox** és **TextBox** vezérlőelemeken akkor következik be, ha a felhasználó felenged egy egérgombot. Az eljárással kapcsolatos tudnivalók értelemszerűen azonosak a **MouseDown** eseménynél leírtakkal.

ObjectMove

Szintaxis: **Private Sub** *objektum*_**ObjectMove**(*bal* **As Single**, *felső* **As Single**, *szélesség* **As Single**, *magasság* **As Single**)

Ez az esemény **OLE** befogadó vezérlőelemen egy benne lévő vezérlés elmozgatása vagy átméretezése után következik be. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *bal* argumentum egy **Single** érték, amely az **OLE** befogadó vezérlőelem bal oldali élének x koordinátáját adja meg, közvetlenül az elmozgatás vagy átméretezés után.
- A *felső* argumentum egy **Single** érték, amely az **OLE** befogadó vezérlőelem felső élének y koordinátáját adja meg, közvetlenül az elmozgatás vagy átméretezés után.
- A *szélesség* argumentum egy **Single** érték, amely az **OLE** befogadó vezérlőelem szélességét adja meg, közvetlenül az elmozgatás vagy átméretezés után.
- A *magasság* argumentum egy **Single** érték, amely az **OLE** befogadó vezérlőelem magasságát adja meg, közvetlenül az elmozgatás vagy átméretezés után.

Amikor a felhasználó mozgat vagy átméretez egy **OLE** befogadó vezérlőelemet, az **ObjectMove** eseménnyel határozható meg az objektum aktuális pozíciója és mérete. Ha az **ObjectMove** eseményeljárás nem változtatja meg az **OLE** befogadó vezérlőelem aktuális pozícióját vagy méretét, az **OLE** befogadó vezérlőelemen belüli objektum az eredeti pozícióját és méretét adja vissza. Az eseménynek argumentumként átadott koordináták az **OLE** befogadó vezérlőelem keretét is magukban foglalják.

Az **ObjectMove** és **Resize** események akkor következnek be, ha az **OLE** befogadó vezérlőelem információt kap az általa tartalmazott objektum méretéről.

Ha az **OLE** befogadó vezérlőelemet elmozgatjuk az űrlapról, az argumentumok pozitív vagy negatív értéke adja meg az objektumnak az űrlap tetejéhez és bal oldalához viszonyított helyzetét.

Paint

Szintaxis 1: **Private Sub Form_Paint( )**

Szintaxis 2: **Private Sub** *objektum*_**Paint(** [*index* **As Integer**] **)**

Ez az esemény a Szintaxis 1 szerint **Form** objektumon, a Szintaxis 2 szerint pedig **PictureBox** vezérlőelemen akkor következik be, ha egy részben vagy teljesen takart űrlap vagy **PictureBox** az elmozgatása, nagyítása vagy az objektumot takaró ablak elmozgatása miatt láthatóvó lesz, és szükségessé válik a takaratlanná váló rész újrarajzolása. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

A **Paint** eseményeljárás a programkódban grafikus metódustól kapott eredmény szükségszerinti újrarajzolására használható. A **Refresh** utasítás automatikusan **Paint** eseményt okoz. Ha a **Refresh** metódust egy **Resize** eseményen belül használjuk, az űrlap minden egyes újraméretezése egyben a teljes objektum újrarajzolását is eredményezi.

Ha az **AutoRedraw** tulajdonság **True**, az újrarajzolás automatikusan, a **Paint** esemény használata nélkül történik. A **ClipControls** tulajdonság **False** értéke esetén a **Paint** csak az újonnan láthatóvá vált területekre hat, **True** beállításnál azonban az űrlap teljes, az **Image**, **Label**, **Line** és **Shape** kivételével bármely más vezérlőelemmel nem fedett része újrarajzolásra kerül.

A **Paint** okozta kaszkád eseménysor bekövetkezésének elkerülésére ne használjuk ezt az eseményeljárást:

- Űrlap vagy vezérlőelem mozgatására vagy méretének módosítására.
- Bármely olyan változó értékének megváltoztatására, amely a méretre vagy megjelenésre hat.
- A **Refresh** metódus meghívására.

A felsorolt műveletek céljára a **Resize** eseményeljárás használata ajánlott.

PathChange

Szintaxis: **Private Sub** *objektum*_**PathChange(** [*index* **As Integer**] **)**

Ez az esemény **FileListBox** vezérlőelemen akkor következik be, ha a programkódból módosítjuk a **FileName** vagy **Path** tulajdonsággal adott elérési utat. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

A **PathChange** eseményeljárás a **FileListBox** vezérlőelemben bekövetkező változás után szükséges fájllista-aktualizálására használható. Minden esetben, amikor a **FileName** vagy **Path** tulajdonságba új elérési utat kijelölő karaktersorozat kerül, egy **PathChange** esemény is bekövetkezik.

PatternChange

Szintaxis: **Private Sub** *objektum*_**PatternChange(** [*index* **As Integer**] **)**

Ez az esemény **FileListBox** vezérlőelemen akkor következik be, ha a programkódból módosítjuk a **FileName** vagy **Pattern** tulajdonsággal adott, például *.* alakú fájllistázási mintát. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

A **PatternChange** eseményeljárás a **FileListBox** vezérlőelemben bekövetkező fájllistázási minta változás után szükséges fájllista-aktualizálására használható. Minden esetben, amikor a **FileName** vagy **Pattern** tulajdonságba új fájllistázási minta kerül, egy **PatternChange** esemény is bekövetkezik.

QueryUnload

Szintaxis 1: **Private Sub Form_QueryUnload(** *elfogadás* **As Integer**, *törlési_ok* **As Integer** **)**

Szintaxis 2: **Private Sub MDIForm_QueryUnload(** *elfogadás* **As Integer**, *törlési_ok* **As Integer** **)**

Ez az esemény **Form** és **MDIForm** objektumon egy űrlap vagy egy alkalmazás bezáródása előtt következik be. Az esemény szintaxisa a következő argumentumokat tartalmazza:

- Az *elfogadás* argumentum egy **Integer** típusú szám, amely nullától különböző értéke esetén az összes betöltött űrlapon megáll a **QueryUnload** esemény, valamint az űrlap és alkalmazás bezárása.
- A *törlési_ok* argumentum egy **Integer** típusú szám vagy egy szimbolikus konstans, amely a **QueryUnload** esemény okát jelöli meg.

Ha egy **MDIForm** objektum záródik be, a **QueryUnload** esemény először az MDI-űrlapon, majd az MDI-alablakokon következik be. Ha űrlap nem érvényteleníti a **QueryUnload** eseményt, az **Unload** esemény elsőként a normál űrlapokon, majd utoljára az MDI-űrlapon következik be. Amikor egy alablak vagy egy **Form** objektum záródik be, az űrlap **Unload** eseménye előtt a **QueryUnload** esemény először azon az űrlapon következik be.

A *törlési_mód* a **QueryUnload** esemény okát a következő lehetőségek közül jelöli ki:

Ha a szimbolikus konstans:	Értéke:	Az esemény oka:
vbFormControlMenu	0	A felhasználó az űrlap Control menüjéből a Close parancsot választotta.
vbFormCode	1	A programkód egy **Unload** metódust hívott.
vbAppWindows	2	Az aktuális Windows környezetben folyó programfutás befejeződött.
vbAppTaskManager	3	A Windows Task Manager bezárta az alkalmazást.
vbFormMDIForm	4	Az MDI-alablak lezárása az MDI-űrlap bezárása miatt következett be.

Az eljárás jellemzően arra használható, hogy az alkalmazás bezárása előtt megbizonyosodjunk arról, hogy az alkalmazás egyik az űrlapján sem maradt befejezetlen feladat. Egy alkalmazás bezáródását a **QueryUnload** vagy **Unload** eseményeljárásban a **Cancel** tulajdonság **True** értékre állításával lehet leállítani. A két esemény között az az eltérés, hogy a **QueryUnload** egy űrlap bezáródásakor az összes űrlapon bekövetkezik, míg az **Unload** mindig csak a bezáródó űrlapon eredményez eseményt.

Reposition

Szintaxis: **Private Sub** *objektum*_**Reposition(** [*index* **As Integer**] **)**

Data vezérlőelemen ez az esemény azután következik be, hogy egy rekord aktuális rekorddá válik. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

Egy **Data** vezérlőelem betöltődésekor annak első rekordja válik aktuális rekorddá, egyben **Reposition** eseményt okozva. Ha a felhasználó az egyik rekordról a másikra lép át, akár a **Data** vezérlőelem egy gombjára kattintva, akár valamelyik **Move** (például a **MoveFirst**, **MoveNext** stb.), **Find** (például **FindFirst**, **FindNext** stb.) vagy más, az aktuális rekordot módosító tulajdonság vagy metódus használatával, a **Reposition** esemény minden egyes új rekord érvényessé válása után bekövetkezik.

A **Reposition** eseményeljárást az aktuális rekord adatával végzett számításokhoz vagy az adatoktól függő űrlapmódosításokhoz lehet felhasználni.

Resize

Szintaxis 1: **Private Sub Form_Resize( )**

Szintaxis 2: **Private Sub** *objektum*_**Resize(** *magasság* **As Single**, *szélesség* **As Single)**

Ez az esemény **Form** és **MDIForm** objektumon, valamint **PictureBox** vezérlőelemen az űrlap első megjelenésekor, illetve a Szintaxis 2 szerinti alakjában a vezérlőelem méretének megváltozásakor következik be. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *magasság* argumentum egy **Single** típusú szám, amely a vezérlőelem új magasságát adja meg.
- A *szélesség* argumentum egy **Single** típusú szám, amely a vezérlőelemet új szélességét adja meg.

A **Resize** eseményeljárást az űrlap méretének megváltozásakor a vezérlőelemek mozgatására, újraméretezésére, továbbá az űrlap méretétől függő változók és tulajdonságok (például **ScaleHeight**, **ScaleWidth** stb.) értékének újraszámolására használhatjuk. Ha az űrlap minden egyes újraméretezésekor a rajta lévő grafikát is méretarányossá kívánjuk tenni, erre a **Resize** eseményeljáráson belül a **Refresh** metódusban hívott **Paint** esemény alkalmas.

Az **AutoRedraw** tulajdonság **False** értéke mellett az űrlap újraméretezése a **Resize**, majd a **Paint** eseményeket is kiváltja, amelyekhez csak kellő elővigyázatosság mellett lehet további eljárásokat adni.

Amikor egy **OLE** befogadó vezérlőelem **SizeMode** tulajdonsága 2 (*Autosize*), a vezérlőelem a **Resize** esemény után automatikusan követi a vezérlőelemben lévő objektum megváltozott méretét. Az ilyen esetben a **Resize** esemény először az objektumra, majd az **OLE** befogadó vezérlőelemre következik be. Az argumentumban a *magasság* és *szélesség* az alkalmazás által meghatározott optimális objektumméretet adja meg. Ha ettől eltérő méretű vezérlőelemre van szükség, a méret módosítása a **Resize** esemény *magasság* és *szélesség* argumentumainak megváltoztatásával lehetséges.

RowColChange

Szintaxis: **Private Sub** *objektum*_**RowColChange(** [*index* **As Integer**, *utolsó_sor* **As String**, *utolsó_oszlop* **As Integer**])

Ezt az eseményt **DBGrid** vagy **Grid** vezérlőelemen az aktuális cella másik cellára változása idézi elő. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A feltételes *utolsó_sor* argumentum a **DBGrid** vezérlőelem előző sorának pozícióját megadó karakteres kifejezés.
- A feltételes *utolsó_oszlop* argumentum a **DBGrid** vezérlőelem előző oszlopának

pozícióját megadó karakteres kifejezés.

A **RowColChange** esemény akkor következik be, ha a vezérlőelem aktuális cellája helyett egy másikat akár az egérrel arra kattintva, akár a programkódban a **Col** és **Row** tulajdonságok használatával aktuálissá teszünk. Nem következik be **RowColChange** esemény, ha a programkódból módosítjuk ugyan a kijelölt tartományt, de az aktuális cella módosítása nélkül.

A **DBGrid** vezérlőelem esetén az aktuális cella pozícióját a **Bookmark** és **ColIndex** tulajdonságok, az előző cella pozícióját pedig az *utolsó_sor* és *utolsó_oszlop* adja meg. Ha az adatot szerkesztjük, majd az aktuális cellapozíciót új sorba visszük át, az eredeti sor frissítése még azelőtt bekövetkezik, hogy az új cella aktuálissá válna.

RowLoaded

Szintaxis: **Private Sub** *objektum*_**RowLoaded(** [*index* **As Integer**,] *könyvjelző* **As String)**

Ez az esemény **DBGrid** vezérlőelemen akkor következik be, amikor a vezérlőelem az adatbázisból betöltődött. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A *könyvjelző* argumentum a betöltött sorhoz tartozó könyvjelzőt megadó karakteres kifejezés.

Az esemény programkódban történő használatával a kötetlen mezők alapértelmezett értékeinek kitöltése vagy a rekord betöltése utáni műveletek végrehajtása lehetséges.

RowResize

Szintaxis: **Private Sub** *objektum*_**RowResize(** [*index* **As Integer**,] *letiltás* **As Integer)**

Ez az esemény a **DBGrid** vezérlőelemen egy **Paint** esemény előtt következik be, amikor a felhasználó sorokat méretez át. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A feltételes *letiltás* argumentum egy **Boolean** típusú kifejezés, amely azt adja meg, hogy a módosítás megtörténjen-e.

A *letiltás* argumentum lehetséges értékei a következő hatásúak:

True Érvényteleníti a módosítást, a sort az eredeti magasságra állítja vissza és érvényteleníti a függőben lévő **Paint** eseményt.

False Folytatja a sor magasságának változtatását és a következő **Paint** eseményt. (Alapértelmezett.)

A felhasználó a **DBGrid** vezérlőelem sorait az egér használatával méretezheti át, és ennek során a magasság változása **RowResize** eseményt okoz. Ennek eseményeljárásában lehetséges a változtatás engedélyezése, mértékének módosítása vagy a változtatás törlése. A magasságváltozás mértékét a **DBGrid** vezérlőelem sorainak magasságát megadó **RowHeight** tulajdonság értékének beállításával lehet befolyásolni.

A **RowResize** eseményeljárásban használt **Refresh** metódus még akkor is újrarajzolja a vezérlőelemet, ha a *letiltás* argumentum **True**.

Scroll

Szintaxis 1: **Private Sub DBGrid_Scroll(** [*letiltás* **As Integer**] **)**

Szintaxis 2: **Private Sub** *objektum*_**Scroll( )**

Ez az esemény **DBGrid**, **HScrollBar** és **VScrollBar** vezérlőelemen akkor következik be, amikor a felhasználó módosítja a gördítősáv csúszkájának helyzetét, illetve függőlegesen vagy vízszintesen gördíti a **DBGrid** vezérlőelemet. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *letiltás* argumentum egy **Boolean** típusú kifejezés, amely azt adja meg, hogy a módosítás megtörténjen-e.

A *letiltás* argumentum lehetséges értékei a következő hatásúak:

True Érvényteleníti a **DBGrid** gördítési műveletet.

False Folytatja az újrarajzolást. (Alapértelmezett.)

Ez az esemény a **DBGrid** vezérlőelemen annak vízszintesen vagy függőlegesen történő elmozdításakor, de még a gördítés eredményének újrarajzolása előtt következik be.

A **Scroll** eseményeljárásban használt **Refresh** metódus még akkor is újrarajzolja a vezérlőelemet az új pozícióban, ha a *letiltás* argumentum **True**, azonban ezt követően az eljárás eredménytelensége miatt egy újabb újrarajzolás következik be az eredeti vezérlőelem-pozícióban. Az eseményeljárásban kerülni kell a **MsgBox** utasítást.

A **Scroll** eseményeljárás a gördítősávokon belüli pozícióval összekapcsolt számításokhoz és vezérlőelem-beállításokhoz használható, ha azonban csak a gördítőcsúszka elmozdulása utáni frissítésre van szükség, erre inkább a **Change** eseményt kell használni.

SelChange

Szintaxis 1: **Private Sub DBGrid_SelChange(** [*letiltás* **As Integer**] **)**

Szintaxis 2: **Private Sub** *objektum*_**SelChange( )**

Ez az esemény **DBGrid** és **Grid** vezérlőelemen akkor következik be, amikor az aktuális tartomány másik cellára vagy cellák tartományára változik. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *letiltás* argumentum egy **Boolean** típusú kifejezés, amely azt adja meg, hogy a módosítás megtörténjen-e.

A *letiltás* argumentum lehetséges értékei a következő hatásúak:

True Érvényteleníti a kijelölési műveletet, a **Grid** vezérlőelemen az aktuális kiválasztást az előző állapotra állítja vissza.
False Folytatja a kijelölést. (Alapértelmezett.)

Ez az esemény akkor következik be, ha a felhasználó az egérrel az aktuálistól különböző cellára kattint, vagy ha egy új cellatartományt jelöl ki. A cellatartomány kijelölése a `SHIFT` billentyű lenyomva tartása mellett a mutatóbillentyűkkel is lehetséges.

Programkódon belül a **SelChange** eseményt a kiválasztott tartomány **SelStartCol**, **SelEndCol**, **SelStartRow** és **SelEndRow** tulajdonságainak, **DBGrid** vezérlőelem esetén pedig a **SelStartCol** és **SelEndCol** tulajdonságainak módosításával idézhetjük elő.

Ha az új cella kiválasztása egérkattintással történik, az egyben egy **SelChange** eseményt is okoz, míg a kijelölésnek a **Grid** vezérlőelemen történő végighúzása vagy a kiválasztott tartomány programból történő, de az aktív cella megváltoztatása nélkül módosítása nem.

Terminate

Szintaxis: **Private Sub** *objektum*_**Terminate( )**

Ez az esemény **ClassModule**, **Form** és **MDIForm** objektumon, valamint **Forms** objektumgyűjteményen következhet be, amikor egy példányukra történő minden hivatkozás a memóriából a rá vonatkozó összes változó **Nothing** értékre beállításával törlésre kerül, vagy ha az objektumra vonatkozó utolsó hivatkozás a hatókörön kívülre esik. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Form** és **MDIForm** objektumon a **Terminate** az **Unload** esemény után következik be.

A **Terminate** esemény nem következik be űrlapra vagy osztályra, ha annak példánya az alkalmazás szabálytalan befejezése — pl. minden létező űrlap vagy osztály példánynak a memóriából törlése előtti **End** utasítás — miatt törlődik a memóriából.

Timer

Szintaxis: **Private Sub** *objektum*_**Timer(** [*index* **As Integer**] **)**

Ez az esemény a **Timer** vezérlőelemen akkor következik be, ha a beállított időköz letelt. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.

Az időköz milliszekundumokban adott értékét a vezérlőelem **Interval** tulajdonsága tartalmazza.

Az eseményeljárásban minden egyes **Timer** időköz lejárta után a Visual Basic által elvégzendő műveleteket helyezhetünk el. A **Timer** esemény használatakor:

- Az **Interval** tulajdonság adja meg két **Timer** esemény közötti időközt milliszekundumban.
- Mindahányszor, ha a **Timer** vezérlőelem **Enabled** tulajdonsága **True**, és az **Interval** tulajdonság nagyobb mint nulla, a **Timer** esemény az **Interval** tulajdonságban adott időközig várakozik.

Unload

Szintaxis: **Private Sub** *objektum*_**Unload(***letiltás* **As Integer)**

Ez az esemény egy **Form** és **MDIForm** objektumon a képernyőről való törlésekor következik be. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *letiltás* argumentum egy **Boolean** típusú kifejezés, amely azt adja meg, hogy a módosítás megtörténjen-e.

A *letiltás* argumentum lehetséges értékei a következő hatásúak:

True Érvényteleníti az űrlap képernyőről való eltávolítását, de nem állít le más eseményeket, így a Windows operációs környezetből történő kilépést sem.

False Folytatja az űrlap eltávolítását. (Alapértelmezett.)

Az **Unload** eseményeljárás az űrlap törölhetőségének ellenőrzésére vagy az űrlap törlésekor szükséges műveletek megadására használható. Ugyancsak tartalmazhat űrlapszintű érvényességi kódot az űrlap bezárásához vagy az adat fájlba mentésére.

A **QueryUnload** esemény az **Unload** esemény előtt, az pedig a **Terminate** esemény előtt következik be.

A következő esetek okoznak **Unload** eseményt:

- az **Unload** utasítás kiadása,
- egy felhasználói művelet, például a Control menü Close tételének kiválasztása,
- a Windows Task Managerben az End Task gomb lenyomása,
- a olyan MDI-űrlap lezárása, amelynek az érvényes űrlap egy alablaka,
- az alkalmazás futása alatt a Windows környezetből való kilépés.

Updated

Szintaxis: **Sub** *objektum*_**Updated(***frissítési_mód* **As Integer)**

Ezt az esemény **OLE** befogadó vezérlőelemen egy adat módosításakor következik be. Az esemény szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *frissítési_mód* argumentum egy szimbolikus konstans vagy **Integer** típusú érték, amely az objektum frissítési módját határozza meg.

A *frissítési_mód* argumentum az adatváltozás módját a következők szerint adja meg:

Ha a szimbolikus konstans:	Értéke:	Az adatváltozás:
vbOLEChanged	0	Az objektum adata megváltozott.
vbOLESaved	1	Az objektum adatát az őt létrehozó alkalmazás mentette.
vbOLEClosed	2	A csatolt objektum adatát tartalmazó fájlt az objektumot létrehozó alkalmazás bezárta.
vbOLERenamed	3	A csatolt objektum adatát tartalmazó fájlt az objektumot létrehozó alkalmazás átnevezte.

Az esemény főként annak meghatározására használható, hogy az objektum adatai az utolsó elmentés óta megváltoztak-e. Ehhez az **Updated** eseménybe egy globális változót kell beállítani, ezzel jelezve, hogy az objektum mentést kíván, majd az objektum mentése után a változó értéke visszaállítható.

Validate

Szintaxis: **Private Sub** *objektum*_**Validate(** [*index* **As Integer**,] *művelet* **As Integer**, *mentés* **As Integer)**

Ez az esemény **Data** vezérlőelemen egy új rekord érvényessé válása előtt következik be, megelőzve az **Update** metódust (kivéve, ha az adatmentés az **UpdateRecord** metó-

dussal történt), valamint a **Delete**, az **Unload** és a **Close** műveleteket. Az esemény szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *index* argumentum egy **Integer** típusú szám, amely a vezérlőelemet akkor jelöli ki, ha az egy tömbben található.
- A *művelet* argumentum egy szimbolikus konstans vagy **Integer** típusú érték, amely az eseményt okozó műveletet határozza meg.
- A *mentés* argumentum a kötött adat változását megadó **Boolean** típusú kifejezés.

A *művelet* argumentum lehetséges értékei a következők:

Ha a szimbolikus konstans:	Értéke:	Az eseményt okozó művelet:
vbDataActionCancel	0	A **Sub** kilépésekor **Cancel**
vbDataActionMoveFirst	1	A **MoveFirst** metódus
vbDataActionMovePrevious	2	A **MovePrevious** metódus
vbDataActionMoveNext	3	A **MoveNext** metódus
vbDataActionMoveLast	4	A **MoveLast** metódus
vbDataActionAddNew	5	Az **AddNew** metódus
vbDataActionUpdate	6	Az **Update** művelet
vbDataActionDelete	7	A **Delete** metódus (nem az **UpdateRecord**)
vbDataActionFind	8	A **Find** (**FindFirst**, **FindPrevious**, **FindNext** vagy **FindLast**) metódus
vbDataActionBookmark	9	A **Bookmark** tulajdonság beállítása
vbDataActionClose	10	A **Close** metódus
vbDataActionUnload	11	Az ablak lezárulása

A *mentés* argumentum lehetséges értékei a következők:

True A kötött adat megváltozott.
False A kötött adat nem változott. (Alapértelmezett.)

A *mentés* argumentum akkor is **False** értékű marad, ha a másolatpuffer megváltozott. Ha az esemény bekövetkeztekor az értéke **True** volt, egy **Edit** és **UpdateRecord** metódus hívódik meg. A kötött vezérlőelemek vagy a másolatpuffer adatai csak a **DataChanged** tulajdonság **True** értéke esetén menthetők az **UpdateRecord** metódussal.

A **Validate** esemény még akkor is bekövetkezhet, ha a kötött vezérlőelemekben lévő adatok nem változtak, vagy kötött vezérlőelemek nem is léteznek.

Az eseményeljárást értékek módosítására, frissítésére, mentésére vagy az eseményt okozó művelet leállítására és másik művelettel történő helyettesítésére használhatjuk. Ez utóbbi a *művelet* argumentum módosításával lehetséges. A műveletek közül a **MoveFirst**, **MovePrevious**, **MoveNext**, **MoveLast** és **AddNew** egymással szabadon cserélhetők, így például az **AddNew** után a **MoveNext**, majd ismét **AddNew** metódus használatával megvizsgálható az **EditMode** tulajdonság, amely alapján meghatározható,

hogy **Edit** vagy **AddNew** művelet van-e folyamatban.

Az **AddNew** és a **Move** metóduscsoport más műveletben történő cseréjének kísérlete vagy figyelmen kívül marad, vagy hibajelzést okoz. Bármely művelet a *művelet* argumentum nullára állításával leállítható.

Az esemény programból történő használatakor először mindegyik adatbázishoz kötött vezérlőelemet megvizsgálva meghatározandó, hogy a **DataChanged** tulajdonság melyiknél **True**, majd azt **False** értékre állítva megakadályozható, hogy az adat az adatbázisba mentődjön.

A **Validate** esemény alatt az alapul szolgáló **Recordset** objektumon semmilyen más metódus sem használható.

Tulajdonságok

AbsolutePosition

Szintaxis: *objektum*.**AbsolutePosition** [= *érték*]

Az **AbsolutePosition** tulajdonság **Recordset** objektumon, valamint dinamikus és pillanatfelvétel típusú **Recordset** objektumokon érvényes az objektum aktuális rekordjának relatív rekordszámának kiolvasására és beállítására. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Long** típusú, 0 (az első rekord) és a **Recordset** objektumban található rekordok száma közötti érték, amellyel egy rekordnak a **Recordset** objektumon belüli helyzetét jelöljük ki. Az objektumban található rekordok számánál nagyobb *érték* beállítása kezelhető hibát okoz.

Az **AbsolutePosition** tulajdonság használatával az aktuális rekord mutatója a rekordhalmaz egy, az *érték* argumentummal kijelölt rekordjára állítható be, illetve a tulajdonság kiolvasásával az aktuális rekord rekordszáma határozható meg.

A **Recordset** objektumban található rekordok számát a **RecordCount** tulajdonság tartalmazza.

Ha a **Recordset** objektumban nincs rekord, így nincs aktuális rekord sem, a tulajdonság értéke -1 lesz. Ha az aktuális rekord törölt, a tulajdonság értéke határozatlan és lekezelhető hiba következik be a rá történő hivatkozáskor.

Minthogy egy rekord pozíciója az azt megelőző valamelyik rekord vagy rekordok törlésekor megváltozik, továbbá a **Recordset** újbóli létrehozásakor — hacsak az nem az **SQL** utasítás **ORDER BY** záradékkal történik — sem szükségszerűen marad változatlan a rekordok sorrendje, az **AbsolutePosition** tulajdonságot nem célszerű a rekordszám helyett használni, egy adott pozíció megőrzésére és az oda visszatérésre a könyvjelzők

használandók.

A tulajdonság nem használható a csak előre gördíthető pillanatfelvétel típusú, illetve utasításátadással megnyitott **Recordset** objektumokon.

Action

Szintaxis:

Az **Action** tulajdonság **CommonDialog** és **OLE** befogadó vezérlőelemen egyaránt érvényes, azonban értelmezésük eltérő.

A tulajdonság a **CommonDialog** vezérlőelemen a megjelenítendő párbeszédpanel típusának beállítására és kiolvasására használható, azonban a Visual Basic 4.0-s verziójában csak kompatibilitási okból maradt meg. Helyette értelemszerűen a **ShowColor**, **ShowFont**, **ShowHelp**, **ShowOpen**, **ShowPrinter** és **ShowSave** metódusokat kell használni.

Az **OLE** befogadó vezérlőelemen a — fejlesztés során nem érvényes — **Action** tulajdonság egy műveletet meghatározó érték beállítására használható, azonban a Visual Basic 4.0-s verziójában csak kompatibilitási okból maradt meg. Helyette értelemszerűen a **Close**, **Copy**, **CreateEmbed**, **CreateLink**, **Delete**, **DoVerb**, **FetchVerbs**, **InsertObjDlg**, **Paste**, **PasteSpecialDlg**, **ReadFromFile**, **SaveToFile**, **SaveToOle1File** és **Update** metódusokat kell használni.

ActiveControl

Szintaxis: *objektum*.**ActiveControl**

Az **ActiveControl** tulajdonság **Form**, **MDIForm** és **Screen** objektumon a fókuszban lévő vezérlőelemet adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. A tulajdonság a programfejlesztési fázisban nem elérhető, a program futása során pedig csak olvasható. Amikor egy űrlapra például a `ChildForm.ActiveControl` alakban hivatkozunk, az **ActiveControl** tulajdonság azt a vezérlőelemet adja meg, amelyik akkor kerülne fókuszba, ha a hivatkozott űrlap aktív lenne. A tulajdonság lekérdezése értelemszerűen hatástalan, ha az űrlap rejtett vagy letiltott.

Az **ActiveControl** tulajdonság arra használható, hogy rajta keresztül elérjünk az éppen aktív vezérlőelem tulajdonságait vagy annak metódusait például a `Screen.ActiveControl.Tag="0"` alakban meghívjuk. Ha a tulajdonság lekérdezésekor az *objektum* mindegyik vezérlőeleme rejtett vagy letiltott állapotú, futásidejű hiba következik be.

Amikor egy vezérlőelem fókuszban van, az egyben a képernyő aktív vezérlőeleme (**Screen.ActiveControl**) is. Ezen túl — függetlenül attól, hogy az adott űrlap aktív-e vagy sem — minden űrlapnak van aktív vezérlőeleme (**Form.ActiveControl**), így az

űrlapok aktív vezérlőelemét még nem aktív űrlapoknál is módosítani lehet a programból.

Az **ActiveControl** tulajdonság különösen MDI-űrlapokon hasznos, ahol az eszköztár minden egyes gombjának egy MDI-alablakbeli vezérlőelemen egy műveletsort kell elindítania, például amikor a felhasználó az eszköztárban a Copy gombra kattint, a programkód az MDI-alablak aktív vezérlőelembeli szövegére az `ActiveForm.ActiveControl.SelText` alakban hivatkozhat.

Ha a **Screen.ActiveControl** értéket egy eljárásnak kívánjuk átadni, az eljárásban az argumentumot valamely meghatározott vezérlőelem típusának (**As TextBox** vagy **As ListBox**) megadása helyett még akkor is általános típusként (**As Control**) kell deklarálni, ha az **ActiveControl** mindig ugyanazon vezérlőelem-típusra hivatkozik.

ActiveForm

Szintaxis: *objektum*.**ActiveForm**

Az **ActiveForm** tulajdonság beépülő **ProjectTemplate**, **MDIForm** és **Screen** objektumon az aktív ablakként szereplő űrlapot adja vissza. Ha egy MDI-űrlap aktív vagy ilyenre történt hivatkozás, az aktív MDI-alűrlapot kapjuk vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az **ActiveForm** tulajdonság arra használható, hogy rajta keresztül elérjünk az éppen aktív vezérlőelem tulajdonságait vagy meghívjuk annak metódusait, például a `Screen.ActiveForm.MousePointer = 4` alakban.

Az **ActiveForm** tulajdonság különösen MDI-űrlapokon hasznos, ahol az eszköztár minden egyes gombjának egy MDI-alablakbeli vezérlőelemen egy műveletsort kell elindítania, például amikor a felhasználó az eszköztárban a Copy gombra kattint, a programkód az MDI-alablak aktív vezérlőelembeli szövegére az `ActiveForm.ActiveControl.SelText` alakban hivatkozhat.

Amikor egy űrlap valamelyik vezérlőeleme fókuszban van, az egyben a képernyő aktív vezérlőeleme (**Screen.ActiveForm**) is. Egy MDI-űrlapnak lehet egy alablaka amely az MDI-űrlap vonatkozásában aktív (**MDIForm.ActiveForm**). Mivel a képernyőn és az MDI-űrlapon érvényes **ActiveForm** szükségszerűen csak akkor azonos, ha egy párbeszédpanel is aktív, az **MDIForm** objektumot az **ActiveForm** tulajdonságként egy párbeszédpanelt megadva is specifikálhatjuk.

Ha egy aktív MDI-alablak nem teljes méretű, a fő- és az alablak címsora egyaránt aktív állapotúként jelenik meg.

Ha a **Screen.ActiveForm** vagy **MDIForm.ActiveForm** értéket egy eljárásnak kívánjuk átadni, az eljárásban az argumentumot valamely meghatározott űrlaptípus megadása helyett még akkor is általános típusként (**As Form**) kell deklarálni, ha az **ActiveForm** mindig ugyanazon űrlaptípusra hivatkozik.

Az **ActiveForm** tulajdonság a **ProjectTemplate** objektumot alapértelmezetten határozza meg.

Align{TM "Align"}

Szintaxis: *objektum*.**Align** [= *érték*]

Az **Align** tulajdonság **Data** és **PictureBox** vezérlőelemekre vonatkozik, és annak kiolvasására vagy beállítására használható, hogy az adott objektum az űrlapon bárhol, vagy csak az űrlap tetejéhez, aljához, jobb vagy bal széléhez igazítottan, automatikusan az űrlap szélességéhez méretezve jelenhet-e meg. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Integer** típusú, az *objektum* megjelenését meghatározó szám.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha az *érték*:	Az objektum megjelenése:
0	*None* — a nem MDI-űrlapok alapbeállítása. A méret és helyzet a programfejlesztési fázisban vagy a programkódból állítható be. Ha az objektum egy MDI-űrlapon van, a beállítás hatástalan.
1	*Top* — az MDI-űrlapok alapbeállítása. Az objektum az űrlap tetején van, szélessége pedig megegyezik az űrlap **ScaleWidth** tulajdonságában adott értékkel.
2	*Bottom* — Az objektum az űrlap alján van, szélessége pedig megegyezik az űrlap **ScaleWidth** tulajdonságában adott értékkel.
3	*Left* — Az objektum az űrlap bal széléhez igazított, szélessége pedig megegyezik az űrlap **ScaleWidth** tulajdonságában adott értékkel.
4	*Right* — Az objektum az űrlap jobb széléhez igazított, szélessége pedig megegyezik az űrlap **ScaleWidth** tulajdonságában adott értékkel.

Ezt a tulajdonságot arra lehet használni, hogy az űrlap alján vagy tetején gyorsan lehessen állapotsort vagy eszköztárat létrehozni. Ha a felhasználó módosítja az űrlap méretét, az **Align** tulajdonságban kijelölt objektum (az érték 1 és 2 beállítása mellett) automatikusan követi a szélességét.

A **Data** és **PictureBox** az egyetlen standard vezérlőelem, amelyet MDI-űrlapon elhelyezhetünk. Az MDI-űrlap felhasználói területe a vezérlőelemekkel nem takart terület, így egy MDI-alűrlap egyetlen vezérlőelemet sem fog takarni, ha azt az MDI-űrlapon teljes méretűvé tesszük.

A tulajdonság *érték* argumentumának 3 és 4 beállításával az eszköztár egy űrlap vagy MDI-űrlap bal vagy jobb éléhez igazítható. Ha egy MDI-űrlap sarkában két eszköztár van, a felülre vagy alulra pozícionált foglalja el a sarkot, kiszorítva onnan a jobb vagy bal élhez igazítottat, amely így az MDI-űrlap belső területét elfoglalva jelenik meg.

Alignment

Szintaxis: *objektum*.**Alignment** [= *érték*]

Az **Alignment** tulajdonság **CheckBox**, **Column**, **DBGrid**, **Label**, **OptionButton** és **TextBox** vezérlőelemekre vonatkozik, és a vezérlőelem helyzetének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Integer** típusú, az *objektum* pozícióját meghatározó szám.

Az *érték* argumentum lehetséges értékei és értelmezésük az egyes vezérlőelemeken a következő:

CheckBox és **OptionButton** vezérlőelemeken:

Ha az *érték*:	A vezérlőelem és a szöveg pozíciója:
0	A vezérlőelem a szöveg jobb oldalára igazított. A szöveg jobbra igazított.(Alapértelmezett.)
1	A vezérlőelem a szöveg bal oldalára igazított. A szöveg jobbra igazított.

Label és **TextBox** vezérlőelemeken:

Ha az érték	A vezérlőelem és a szöveg pozíciója:
0	A szöveg jobbra igazított. (Alapértelmezett.)
1	A szöveg balra igazított.
2	A szöveg középre igazított.

DBGrid oszlopokon:

Ha az *érték*:	A vezérlőelem és a szöveg pozíciója:
0	A szöveg jobbra igazított.
1	A szöveg balra igazított.
2	A szöveg középre igazított.
3	A szöveg balra, a számok jobbra igazítottak. (Alapértelmezett.)

Ha a **TextBox** vezérlőelem **MultiLine** tulajdonsága **False**, az **Alignment** tulajdonság értéke közömbös.

AllowAddNew

Szintaxis: *objektum*.**AllowAddNew** [= *érték*]

Az **AllowAddNew** tulajdonság **DBGrid** vezérlőelemekre vonatkozik, és a vezérlő-

elem alapjául szolgáló **Recordset** objektumhoz történő új rekord hozzáadásának lehetőségét jelző **Boolean** érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A felhasználó rekordokat adhat a **DBGrid** alapjául szolgáló **Recordset** objektumhoz.

False A felhasználó nem adhat rekordokat a **DBGrid** alapjául szolgáló Recordset objektumhoz.

A **DBGrid** vezérlőelemben az utolsó megjelenített sor üresen marad, így a felhasználónak lehetőséget ad egy új rekord hozzáadására. Ha az **AllowAddNew** tulajdonság **False** értékű, a felhasználó nem viheti a fókuszt az üres sorra.

Lehetséges, hogy a vezérlőelem alapjául szolgáló **Recordset** még akkor sem engedi meg új rekord hozzáadását, ha a **DBGrid** vezérlőelemének **AllowAddNew** tulajdonsága **True** értékű. Az ilyen esetben a felhasználó által megkísérelt rekordhozzáadás hibát okoz.

AllowDelete

Szintaxis: *objektum*.**AllowDelete** [= *érték*]

Az **AllowDelete** tulajdonság **DBGrid** vezérlőelemekre vonatkozik, és a vezérlőelem alapjául szolgáló **Recordset** objektumból történő rekordtörlés lehetőségét jelző **Boolean** érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A felhasználó rekordokat törölhet a **DBGrid** alapjául szolgáló **Recordset** objektumhoz.

False A felhasználó nem törölhet rekordokat a **DBGrid** alapjául szolgáló **Recordset** objektumhoz.

Az **AllowDelete** tulajdonság használatával megakadályozható, hogy a felhasználó a **DBGrid** vezérlőelemen keresztül rekordokat tudjon törölni a **Recordset** objektumból.

Lehetséges, hogy a vezérlőelem alapjául szolgáló **Recordset** még akkor sem engedi meg rekord törlését, ha a **DBGrid** vezérlőelemének **AllowDelete** tulajdonsága **True** értékű. Az ilyen esetben a felhasználó által megkísérelt rekordtörlés hibát okoz.

AllowRowSizing

Szintaxis: *objektum*.**AllowRowSizing** [= *érték*]

Az **AllowRowSizing** tulajdonság **DBGrid** vezérlőelemekre vonatkozik, és a vezérlőelem sorainak átméretezési lehetőségét jelző **Boolean** érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A felhasználó átméretezheti a sorokat.
False A felhasználó nem méretezheti át a sorokat.

Ha az **AllowRowSizing** tulajdonság értéke **True**, az egér mutatóját minden rekord közötti osztóvonal felett kettős (Size N S) nyílra változik, és a felhasználó elhúzással módosíthatja a sorok magasságát. A sorméret bármely változtatása **RowResize** eseményt okoz.

A **DBGrid** vezérlőelem mindegyik sorának magassága azonos, amelyet a **RowHeight** tulajdonság értéke határoz meg.

AllowSizing

Szintaxis: *objektum*.**AllowSizing** [= *érték*]

Az **AllowSizing** tulajdonság **Column** vezérlőelemekre vonatkozik, és a **DBGrid** vezérlőelem adott oszlopának átméretezési lehetőségét jelző **Boolean** érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A felhasználó átméretezheti az oszlopot.
False A felhasználó nem méretezheti át az oszlopot.

Ha az **AllowSizing** tulajdonság értéke **True**, az egér mutatóját egy adott oszlop osztóvonala felett kettős (Size W E) nyílra változik, és a felhasználó elhúzással módosíthatja az oszlop méretét. Az oszlopméret bármely változtatása **ColResize** eseményt okoz.

A **DBGrid** vezérlőelem mindegyik sorának magassága azonos, amelyet a **RowHeight** tulajdonság értéke határoz meg.

AllowUpdate

Szintaxis: *objektum*.**AllowUpdate** [= *érték*]

Az **AllowUpdate** tulajdonság **DBGrid** vezérlőelemekre vonatkozik, és a **DBGrid** vezérlőelemben az adatok megváltoztatásának lehetőségét jelző **Boolean** érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A felhasználó módosíthatja a **DBGrid** vezérlőelem adatait.
False A felhasználó nem módosíthatja a **DBGrid** vezérlőelem adatait.

Ha az **AllowUpdate** tulajdonság értéke **False**, a felhasználó a **DBGrid** vezérlőelemen tetszés szerint mozoghat és adatot is kiválaszthat, azonban az értéket nem változtathatja meg. Az adatváltoztatás kísérlete hibát okoz.

A **Column** objektum tulajdonságai a **DBGrid** vezérlőelem egyes oszlopainak csak olvashatóvá tételére is használhatók, azonban figyelemmel kell lenni arra, hogy az **AllowUpdate** tulajdonságnak mindig elsőbbsége van az oszlop beállításaihoz viszonyítva, noha azokat változatlanul hagyja.

Lehetséges, hogy a vezérlőelem alapjául szolgáló **Recordset** még akkor sem engedi meg a frissítést, ha a **DBGrid** vezérlőelemének **AllowDelete** tulajdonság **True** értékű. Az ilyen esetben a felhasználó által megkísérelt rekordmódosítás kezelhető hibát okoz.

AllowZeroLength

Szintaxis: *objektum*.**AllowZeroLength** [= *érték*]

Az **AllowZeroLength** tulajdonság **Field** objektumon érvényes, és a **Text** vagy **Memo** adattípusú objektum **Value** tulajdonságában az üres ("") karakterlánc érvényes beállításkénti érvényességét jelző **Boolean** érték kiolvasására vagy beállítására használható. **Fields** objektumgyűjteményhez még nem tartozó objektum esetén a tulajdonság írható és olvasható is, egyébként használata az objektumgyűjteményt tartalmazó objek-

tumtól függ. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True	Az üres ("") karakterlánc érvényes érték.
False	Az üres ("") karakterlánc nem érvényes érték.

Az **AllowZeroLength** tulajdonság használhatóságát a **Field** objektum **Fields** halmazát tartalmazó objektum a következők szerint határozza meg:

Ha az objektum:	A tulajdonság használata:
Index	Nem támogatott.
QueryDef	Csak olvasható.
Recordset	Csak olvasható.
Relation	Nem támogatott.
TableDef	Írható és olvasható.

Az **AllowZeroLength** tulajdonság értékeként **False** értéket beállítva biztosítható, hogy a **Field** objektum **Value** értékeként ne lehessen üres karakterláncot használni, a **Required**, **ValidateOnSet** és **ValidationRule** tulajdonságokkal együtt pedig egy mezőbeli érték érvényességének ellenőrzésére használható.

AllPermissions

Szintaxis: *objektum*.**AllPermissions**

Az **AllPermissions** tulajdonság **Container** és **Document** objektumon az objektum aktuális **UserName** tulajdonságában érvényes jogosultságokat adja vissza, beleértve a kizárólag a felhasználóra vonatkozó és egy csoport tagjaként érvényes jogosultságokat is. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Container** és **Document** objektumon a visszakapott szimbolikus konstansok és értelmezésük a következő lehet:

Ha a szimbolikus konstans:	Értelmezése:
dbSecReadDef	Jogosultság a tábladefiníció (beleértve az oszlop és index információkat is) olvasására.
dbSecWriteDef	Jogosultság a tábladefiníció (beleértve az oszlop és index információkat is) módosítására vagy törlésére.
dbSecRetrieveData	Jogosultság adatok **Document** objektumból történő kiolvasására.

dbSecInsertData	Jogosultság rekordok hozzáadására.
dbSecReplaceData	Jogosultság rekordok módosítására.
dbSecDeleteData	Jogosultság rekordok törlésére.

A **Database** befogadó vagy a **Documents** halmaz bármely **Document** objektumán érvényes további szimbolikus konstansok és értelmezésük a következő lehet:

Ha a szimbolikus konstans:	Értelmezése:
dbSecDBAdmin	A felhasználó jogosult egy adatbázis másolhatóvá tételére és jelszavának módosítására.
dbSecDBCreate	A rendszer adatbázisában (SYSTEM.MDW) lévő **Database Container** objektum esetén jogosultság új adatbázis létrehozására.
dbSecDBExclusive	Kizárólagos hozzáférési jogosultság.
dbSecDBOpen	Adatbázis megnyitási jogosultság.

Az **AllPermissions** tulajdonsággal szemben a **Permission** tulajdonság csak a felhasználóra jellemző jogosultságot adja vissza, a felhasználó egy csoport tagjakénti jogosultságait nem. Ha a **UserName** egy csoport, az **AllPermissions** és a **Permission** tulajdonság értéke azonos.

Appearance

Szintaxis: *objektum*.**Appearance** [= *érték*]

Az **Appearance** tulajdonság **CheckBox**, **ComboBox**, **CommandButton**, **Data**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox** és **TextBox** vezérlőelemeken, valamint **Form** és **MDIForm** objektumon érvényes, és a **Form** vagy **MDIForm** objektumon lévő vezérlőelemek futásidőben érvényes rajzolási stílusának kiolvasására vagy beállítására használható. A futásidőben csak olvasható tulajdonság. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Integer** kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha az *érték*:	Értelmezése:
0	Sík — az űrlapok és vezérlőelemek rajzolása vizuális hatások nélkül történik.
1	3D — az űrlapok és vezérlőelemek rajzolása háromdimenziós hatással történik.

Az **Appearance** *érték* argumentumot programfejlesztési időben 1-re beállítva a vezérlőelemek háromdimenziós hatásúak lesznek. Ugyanezen beállítás egy **MDIObject** objektumon csak az MDI-űrlapra hat, az MDI-alablakok háromdimenziós hatásúvá tételéhez minden egyes alablak **Appearance** tulajdonságát is ugyanígy kell beállítani.

Ha az **Appearance** *érték* argumentum értéke 1, az űrlap és vezérlőelemek **BackColor** tulajdonsága azt a színt kapja, amely az operációs rendszer Control Panel Color beállításának Button Face színeként van megadva.

Ha egy űrlap **BorderStyle** tulajdonságának értéke 3 (vagy **vbFixedDouble**), az űrlap címe és határa ugyancsak háromdimenziós hatásúvá válik.

Az OLE vezérlőelemek 16-bites verziója nem tartalmazza az **Appearance** tulajdonságot.

AppIsRunning

Szintaxis: *objektum*.**AppIsRunning** [= *érték*]

Az **AppIsRunning** tulajdonság **OLE** befogadó vezérlőelemekre vonatkozik, és a vezérlőelemben az objektumot létrehozó alkalmazás futását jelző **Boolean** érték kiolvasására vagy beállítására használható. A programfejlesztési fázisban nem használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True Az OLE befogadó vezérlőelemben az objektumot létrehozó alkalmazás fut.
False Az OLE befogadó vezérlőelemben az objektumot létrehozó alkalmazás nem fut.

Az **OLE** befogadó vezérlőelemben az objektumot létrehozó alkalmazás futása az **AppIsRunning** tulajdonság **True** beállításával indítható el, ami az objektumok gyorsabb aktiválását is eredményezi. Ha az objektum elveszti a fókuszt, az alkalmazás bezárása a tulajdonság **False** beállításával történhet.

Archive

Szintaxis: *objektum*.**Archive** [= *érték*]

Az **Archive** tulajdonság **FileListBox** vezérlőelemekre vonatkozik, és a vezérlőelemben listázandó fájlok **Archive** attribútumát beállító **Boolean** érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True	A **FileListBox** vezérlőelemben az Archive attribútumú fájlokat jeleníti meg. (Alapértelmezett)
False	A **FileListBox** vezérlőelemben a nem Archive attribútumú fájlokat jeleníti meg.

A tulajdonsággal a **FileListBox** vezérlőelemben megjelenítendő, az operációs környezetben használt standard fájlattribútum állítható be. Ha a tulajdonság beállítása futásidőben, programkódból történik, a **FileListBox** vezérlőelem újra beállítódik és csak az adott attribútumú fájlok jelennek meg.

Attributes

Szintaxis: *objektum*.**Attributes** [= *érték*]

Az **Attributes** tulajdonság **Field**, **Relation** és **TableDef** objektumokra vonatkozik, és az objektum egy vagy több jellemzőjének kiolvasására vagy beállítására használható. Objektumgyűjteményhez még nem tartozó objektum esetén a tulajdonság írható és olvasható is, egyébként használata az objektumgyűjteményt tartalmazó objektumtól függ. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy szimbolikus konstans.

A **Field** objektumon az objektum által képviselt mező jellemzőit a következő szimbolikus konstansok összege határozhatja meg:

Ha a szimbolikus konstans:	Értelmezése:
dbFixedField	A mező mérete rögzített hosszúságú. Numerikus mezőkön alapértelmezett.
dbVariableField	A mező mérete változó. Csak Text mezőkön érvényes.
dbAutoIncrField	Az új rekordokhoz tartozó mezőérték automatikusan egy **Long** típusú, nem módosítható egyedi értékkel nő. Csak a Microsoft Jet adatbázis táblák támogatják.
dbUpdatableField	A mezőérték módosítható.
dbDescending	A mezők fordított sorrendbe (Z-A vagy 100-0) rendezettek. Csak egy **Index** objektum **Fields** objektumgyűjteményében lévő **Field** objektumon használható. Ha a konstans nem szerepel az *érték* argumentumban, a rendezés a normál sorrendnek megfelelő lesz.

Az **Attributes** tulajdonság használhatóságát a **Field** objektum **Fields** halmazát tartalmazó objektum a következők szerint határozza meg:

Ha az objektum:	A tulajdonság használata:
Index	Csak olvasható.
QueryDef	Csak olvasható.
Recordset	Csak olvasható.
Relation	Nem támogatott.
TableDef	Csak olvasható.

A **Relation** objektumon az objektum által képviselt reláció jellemzőit a következő szimbolikus konstansok összege határozhatja meg:

Ha a szimbolikus konstans:	Értelmezése:
dbRelationUnique	A reláció egy-az-egyhez.
dbRelationDontEnforce	A reláció nem kényszerített, vagyis nincs hivatkozási integritás.
dbRelationInherited	A reláció a két hozzáfűzött táblát tartalmazó, nem aktuális adatbázisban áll fenn.
dbRelationLeft	A reláció egy bal csatlakozás.
dbRelationRight	A reláció egy jobb csatlakozás.
dbRelationUpdateCascade	A frissítés kaszkád.
dbRelationDeleteCascade	A törlés kaszkád.

Ha az **Attributes** tulajdonság kaszkád műveleteket aktiváló beállítású, a Microsoft Jet az elsődleges kulcstáblával kapcsolatos változtatás esetén a rekordokat egy vagy több másik táblán automatikusan frissíti vagy törli.

Az **Attributes** tulajdonság egy halmazhoz adott **Relation** objektum esetén csak olvasható.

A **TableDef** objektumon az objektum által képviselt tábla jellemzőit a következő szimbolikus konstansok összege határozhatja meg:

Ha a szimbolikus konstans:	Értelmezése:
dbAttachExclusive	A Microsoft Jetet használó adatbázisnál azt jelzi, hogy a tábla egy csatolt tábla és kizárólagos használatra van megnyitva.
dbAttachSavePWD	A Microsoft Jetet használó adatbázisnál azt jelzi, hogy csatolt tábla felhasználói azonosítója és jelszava a kapcsolatleírással együtt van mentve.
dbSystemObject	Azt jelzi, hogy a tábla a Microsoft Jet által szolgáltatott rendszertábla. (Csak olvasható.)
dbHiddenObject	Azt jelzi, hogy a tábla a Microsoft Jet által szolgáltatott időleges, rejtett tábla. (Csak olvasható.)

dbAttachedTable	Azt jelzi, hogy a tábla nem ODBC adatbázisból (MS Access, Paradox) csatolt tábla.
dbAttachedODBC	Azt jelzi, hogy a tábla ODBC-adatbázisból (SQL Server, Oracle) csatolt tábla.

Az **Attributes** tulajdonság használata a **TableDef** objektum állapotától a következők szerint függ

Ha a **TableDef**:	Az **Attributes**:
Alaptábla	Csak olvasható.
Csatolt tábla	Csak olvasható.

Az **Attributes** tulajdonság egy adott attribútumának vizsgálatához az **And** logikai műveletet használhatjuk, például: `MezoNev.Attributes And dbAttachedODBC` alakban.

Többszörös attribútumok beállítása a konstansok összegzésével lehetséges, például a `MezoNev.Attributes = dbVariableField + dbUpdatableField` alakban. Az adott objektum esetén nem értelmezett konstansok hibajelzés nélkül figyelmen kívül maradnak.

AutoActivate

Szintaxis: *objektum*.**AutoActivate** = [*érték*]

Az **AutoActivate** tulajdonság **OLE** befogadó vezérlőelemen egy objektumnak a vezérlőelemre duplán kattintva vagy a fókuszt a vezérlőelemre ráállítva történő aktiválásának lehetőségét jelző szimbolikus konstans vagy **Integer** típusú érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy szimbolikus konstans vagy **Integer** érték.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	Értéke:	Az objektum aktiválási módja:
vbOLEActivateManual	0	*Manual* — Az OLE objektum nem automatikusan aktiválódik, ez a programból a **DoVerb** metódussal lehetséges.

vbOLEActivateGetFocus	1	*Focus* — Ha az **OLE** befogadó vezérlőelem az egyetlen kattintást támogató objektumot tartalmaz, az objektumot szolgáltató alkalmazás akkor aktiválódik, amikor az **OLE** befogadó vezérlőelem megkapja a fókuszt.
vbOLEActivateDoubleclick	2	*Double-Click* — Ha az **OLE** befogadó vezérlőelem objektumot tartalmaz, az azt szolgáltató alkalmazás akkor aktiválódik, amikor a felhasználó duplán kattint az OLE vezérlőelemre vagy akkor üti le az ENTER billentyűt, amikor a vezérlőelem fókuszban van. (Alapértelmezett.)
vbOLEActivateAuto	3	*Automatic* — Ha az **OLE** befogadó vezérlőelem objektumot tartalmaz, az azt szolgáltató alkalmazás az objektum szokásos módja szerint, azaz vagy akkor aktiválódik, amikor a felhasználó duplán kattint az OLE vezérlőelemre vagy ha akkor üti le az ENTER billentyűt, amikor a vezérlőelem fókuszban van.

Az **OLE** befogadó objektum **OLEType** tulajdonságát ellenőrizve határozható meg, hogy az tartalmaz-e objektumot.

Ha az **AutoActivate** tulajdonság értéke 2 (*Double-Click*), a felhasználó által egy **OLE** befogadó objektumra történő dupla kattintás nem okoz **DblClick** esemény.

AutoRedraw

Szintaxis: *objektum*.**AutoRedraw** = [*érték*]

Az **AutoRedraw** tulajdonság **Form** objektumon és **PictureBox** vezérlőelemen egy grafikus metódus állandó grafikai kimenetének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

A **Boolean** típusú *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True	Engedélyezi a **Form** objektum vagy **PictureBox** vezérlőelem újrarajzolását. A grafika és a szöveg a képernyőre kerül és a képmemóriában tárolódik. Az objektum nem kapja meg a **Print** eseményeket, az újrarajzolás pedig a képmemória tartalmából szükség szerint történik.

False Letiltja az objektum automatikus újrarajzolását és a grafika vagy szöveg csak a képernyőre kerül. Az újrarajzoláshoz meghívódik az objektumhoz tartozó **Paint** esemény. (Alapértelmezett.)

Ez a tulajdonság elsősorban a grafikus utasításokkal (**Circle**, **Cls**, **Line**, **Point**, **Print** és **PSet**) történő munkákhoz kötődik. Az **AutoRedraw** tulajdonság **True** értéke esetén automatikusan újrarajzolódik ezen metódusok eredménye egy **Form** objektumban vagy **PictureBox** vezérlőelemben, például az objektum átméretezésekor vagy rejtett állapotból történő újbóli megjelenítésekor.

Az **AutoRedraw** beállítással a programból futásidőben lehet az állandó (például háttér vagy rács) és az időleges grafikák megjelenítése között átkapcsolni. Az újrarajzolás **False** értékű beállításával történő letiltásakor az eddigi kimenet a háttérrajz részévé válik, így a **Cls** metódussal a tulajdonság **True** értékre való visszaállításáig nem lesz törölhető.

A **BackColor** tulajdonság van beállításakor minden grafika és szöveg, így az állandó háttérrajz is törlődik. Általában minden grafikát a **Paint** eseményeljárásban kívánatos elhelyezni, kivéve, ha az **AutoRedraw** tulajdonság **True**. Az ilyen beállításban a bitképes háttérrajzok visszaállítása az **Image** tulajdonság beállításával, a Windows API részére történő átadása pedig az objektum **hDC** tulajdonságával lehetséges.

Ha az **AutoRedraw** tulajdonság **False** beállítása után az űrlapot ikonállapotúvá tesszük, a **ScaleHeight** és **ScaleWidth** az ikon méretének megfelelő értékű lesz, míg az **AutoRedraw** tulajdonság **True** beállításnál megtartják a visszaállított ablak méretét.

AutoShowChildren

Szintaxis: *objektum*.**AutoShowChildren** = [*érték*]

Az **AutoShowChildren** tulajdonság **MDIForm** objektumon érvényes, és az MDI-alablak betöltéskori automatikus megjelenítését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

A **Boolean** típusú *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True Az MDI-alablak betöltéskor automatikusan megjelenített lesz. (Alapértelmezett.)

False Az MDI-alablak betöltéskor nem lesz automatikusan megjelenített.

Ezt a tulajdonság elsősorban arra használható, hogy az MDI-alablak rejtetten lehessen betölteni, majd a kívánt pillanatban a **Show** metódussal megjeleníteni.

AutoSize

Szintaxis: *objektum*.**AutoSize** = [*érték*]

Az **AutoSize** tulajdonság **Label** és **PictureBox** vezérlőelemen érvényes, és vezérlőelem méretének a tartalomhoz megfelelő automatikus átméretezését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

A **Boolean** típusú *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True A vezérlőelem a teljes tartalom megjelenítéséhez szükséges mértékben automatikusan átméreteződik.

False A vezérlőelem mérete állandó marad és a határán túlnyúló tartalom elvész. (Alapértelmezett.)

AutoVerbMenu

Szintaxis: *objektum*.**AutoVerbMenu** = [*érték*]

Az **AutoVerbMenu** tulajdonság **OLE** befogadó vezérlőelemen érvényes, és az azt meghatározó érték kiolvasására vagy beállítására használható, hogy a felhasználónak a JOBB egérgombbal a vezérlőelemre kattintásakor megjelenjen-e az objektum parancsait tartalmazó előugró menü. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

A **Boolean** típusú *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True Az OLE vezérlőelemre JOBB egérgombbal történő kattintás megjeleníti az objektum által támogatott parancsok előugró menüjét. (Alapértelmezett.)

False Nem jelenít meg előugró menüt.

Az **AutoVerbMenu** tulajdonság **True** beállításánál a JOBB gombbal történő kattintás nem eredményez **Click** és **MouseDown** eseményeket.

BackColor

Szintaxis: *objektum*.**BackColor** = [*érték*]

A **BackColor** tulajdonság **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape** és **TextBox** vezérlőelemeken, valamint **Column**, **Form** és **MDIForm** objektumon érvényes, és az objektum háttérszínének kiolvasására vagy beállítására használható.

A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a háttérszínt meghatározó konstans vagy színkód értéke.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
Normál RGB szín	A Color paletta vagy a programkódban az **RGB** vagy **QBColor** függvénnyel meghatározott szín.
Alapértelmezett szín	A Visual Basic objektumtárban a rendszer színkonstansával meghatározott szín. A Windows környezet a felhasználó színválasztását a Control Panel érvényes színével helyettesíti.

A programfejlesztési fázisban minden űrlapra és vezérlőelemre a **vbWindowBackground** konstans szerinti háttérszín lesz érvényes.

A **Label** és **Shape** vezérlőelemekben a **BackStyle** tulajdonság 0 (*Transparent*) értéke esetén a **BackColor** tulajdonság hatástalan.

A **BackColor** tulajdonságot **Form** objektumon vagy **PictureBox** vezérlőelemen beállítva, minden szöveg vagy grafika (az állandó grafikákat is beleértve) törlődik. Az egyéb vezérlőelemeken a képernyőszín azonnal követi a beállításokat.

A normál RGB színeket meghatározó *érték* számértékének érvényes tartománya 0 és 16777215 (**&HFFFFFF**) között lehet. A felső byte értéke ebben a tartományban 0, az alsó bitcsoport a legalacsonyabb bittől felfelé a vörös, zöld és kék színösszetevőt határozza meg. Ezen összetevők 0 és 255 (**&HFF**) közötti értékek lehetnek. Ha a felső byte nem nulla, a Visual Basic a Control Panel beállításának és az Object Browser objektumtárában található konstansoknak megfelelő rendszerszíneket használja.

Windows környezetben szöveg megjelenítéséhez szöveg- és háttérszínként egyaránt tiszta színeket kell használni. Ha a szöveg- és háttérszín valamelyikeként tiszta szín helyett egy három színből álló színmintát választottunk, az a hozzá legközelebbi tiszta színnel helyettesítődik.

BackStyle

Szintaxis: *objektum*.**BackStyle** = [*érték*]

A **BackStyle** tulajdonság **Label**, **OLE** befogadó és **Shape** vezérlőelemeken érvényes, és a **Label** vezérlőelemnek vagy a **Shape** vezérlőelem hátterének átlátszóságát meghatározó érték kiolvasására vagy beállítására használható.

A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, az átlátszóságot meghatározó konstans vagy színkód értéke.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
0	*Transparent* — A vezérlőelem mögött a háttérszín és minden grafika látható, a vezérlőelem **BackColor** tulajdonsága érvénytelen.
1	*Opaque* — A vezérlőelem **BackColor** tulajdonságában adott szín kitölti a vezérlőelemet, így takarva a mögötte lévő színt vagy grafikát. (Alapértelmezett.)

A **BackStyle** tulajdonság használatával egy vezérlőelem a mögötte lévő területet fedő mód helyett úgy is megjeleníthető, hogy a mögötte lévő területen bekövetkező változásokat is figyelemmel kísérhessük, ha a **Form** objektumon vagy **PictureBox** vezérlőelemen háttérszínt használunk vagy grafika fölé vezérlőelemet helyeztünk.

BOF

Szintaxis: *objektum*.**BOF**

A **BOF** tulajdonság **Recordset** (valamint az elavult **Dynaset**, **Snapshot** és **Table**) objektumon érvényes, és az aktuális rekordnak egy objektum első rekordja előtti pozícióját jelző értéket ad vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság által visszaadott, az aktuális rekord mutatójától függő **Boolean** érték értelmezése a következő:

Ha a **BOF** értéke:	Értelmezése:
True	Az érvényes rekord pozíciója az első rekord előtt van.
False	A rekord az első vagy azt követő bármelyik pozícióban van.

A **BOF** (és **EOF**) tulajdonságot annak meghatározására lehet használni, hogy egy **Recordset** objektum tartalmaz-e rekordot, illetve hogy egy **Recordset** objektumban rekordról rekordra lépve túlléptük-e annak határait.

Ha rekordot nem tartalmazó **Recordset** objektumot nyitunk meg, a **BOF** (és **EOF**) tulajdonság értéke **True**, a rekordhalmaz **RecordCount** tulajdonsága pedig 0 lesz. Ha legalább egy rekordot tartalmazó **Recordset** van megnyitva, az aktuális rekord a halmaz első rekordja lesz, a **BOF** tulajdonság értéke pedig mindaddig **False** marad, ameddig a **MovePrevious** metódussal az első rekord elé nem akarunk lépni. Ilyen esetben nem lesz aktuális rekord sem kijelölve.

A **BOF** és **EOF** tulajdonságok különböző kombinációi esetén a megengedett **Move** metódusokat a következő táblázat foglalja össze.

Ha BOF és EOF	MoveFirst, MoveLast	MovePrevious, Move <0	Move 0	MoveNext, Move > 0
BOF =True, EOF = False	Megengedett	Hibát okoz	Hibát okoz	Megengedett
BOF =False, EOF = True	Megengedett	Megengedett	Hibát okoz	Hibát okoz
BOF =True, EOF = True	Hibát okoz	Hibát okoz	Hibát okoz	Hibát okoz
BOF =False, EOF = False	Megengedett	Megengedett	Megengedett	Megengedett

Ha egy **Recordset** objektum utolsó rekordját is töröljük, a **BOF** (és **EOF**) tulajdonság az aktuális rekord újrapozícinálási kísérletéig **False** értékű marad.

Ha egy rekordot tartalmazó **Recordset** objektumon a halmaz első rekordját aktuális rekorddá tevő **MoveFirst** metódus után a **MovePrevious** metódust használjuk, az aktuális rekord érvénytelenné válik és a **BOF** értéke **True** lesz. Ha ilyenkor ismét használjuk a **MovePrevious** metódust, hiba következik be.

A táblázatban megengedettnek jelölt metódusok használata nem okoz hibajelzést, de nem szükségszerűen eredményezi rekord sikeres kijelölését. A rekordot nem kijelölő **Move** esetében a **BOF** tulajdonság értéke a következő lesz:

	MoveFirst, MoveLast	MovePrevious, Move <0	Move 0	MoveNext, Move > 0
BOF értéke	True	True	Változatlan	Változatlan

A **BOF** tulajdonság értékét még a következő metódusok is módosíthatják:

- Mivel az **OpenRecordset** metódus egy **MoveFirst** metódust is meghív, így üres rekordhalmazon a **BOF** tulajdonságot **True** értékre állítja be.
- Minden sikeresen rekordot meghatározó **Move** metódus a **BOF** tulajdonságot **False** értékre állítja be.
- Egy **AddNew** metódust követő, az új rekordot sikeresen beillesztő **Update** metódus **False** értékre állítja a **BOF** tulajdonság értékét, de csak akkor, ha az **EOF** értéke **True** volt. A Microsoft Jet által definiáltak szerint egy üres rekordhal-

mazban az aktuális rekord mutatója egy fájl végénél van, így minden új rekord az aktuális rekord utáni pozícióba kerül.

- A **Delete** metódus, — még akkor is, ha a rekordhalmaz utolsó rekordját törli — változatlanul hagyja a **BOF** tulajdonság értékét.

BOFAction

Szintaxis: *objektum*.**BOFAction** = [*érték*]

A **BOFAction** tulajdonság **Data** vezérlőelemen érvényes, és a **BOF** tulajdonság **True** értéke esetén bekövetkező műveletet megadó érték kiolvasására vagy beállítására használható.

A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a műveletet meghatározó szimbolikus konstans vagy **Integer** érték.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	A művelet:
vbBOFActionMoveFirst	0	**MoveFirst** — Az első rekord marad az aktuális rekord. (Alapértelmezett.)
vbBOFAActionBOF	1	**BOF** — A **Recordset** elejének elhagyása a **Data** vezérlőelem **Validate** eseményét okozza az első rekordon, amit egy **Reposition** esemény követ az érvénytelenné vált (**BOF**) rekordon. Ekkor a **Data** vezérlőelem Move Previous gombja nem elérhetővé válik.

Bold

Szintaxis: *objektum*.**Bold** = [*érték*]

A **Bold** tulajdonság **Font** objektumon érvényes, és az objektum félkövér vagy nem félkövér megjelenését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

A **Boolean** típusú *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True A **Font** objektum megjelenítése félkövér lesz.
False A **Font** objektum megjelenítése nem lesz félkövér. (Alapértelmezett.)

Mivel a **Font** objektum a programfejlesztési fázisban közvetlenül nem érhető el, **Bold** tulajdonságának értékét a kívánt vezérlőelem Properties ablakban a Font tulajdonság kiválasztásával és a Properties gombra kattintással, majd a Font párbeszédpanel Font Style mezőjében a Bold vagy Bold Italic kiválasztásával állíthatjuk be. A tulajdonság futásidőben történő beállítása közvetlenül, a **Font** objektum számára a szükséges értéket megadva történhet.

Bookmark

Szintaxis: *objektum*.**Bookmark** = [*érték*]

A **Bookmark** tulajdonság **Recordset** (valamint az elavult **Dynaset**, **Snapshot** és **Table**) objektumon és **DBGrid** vezérlőelemen érvényes, és a **Recordset** objektum aktuális rekordját azonosító könyvjelző kiolvasására vagy beállítására, illetve a **Recordset** objektumban aktuális rekordhoz egy érvényes könyvjelző beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **String** vagy **Variant** típusú, könyvjelzőként értelmezett kifejezés.

A **Bookmark** tulajdonság alkalmazása előtt ellenőrizni kell, hogy az adott **Recordset** támogatja-e a könyvjelző használatát vagy sem. Ez utóbbi esetet a **Bookmarkable** tulajdonság **False** értéke jelzi és a tulajdonság használata kezelhető hibát okoz. A **Bookmarkable** tulajdonság egy eredetileg Microsoft Jet adatbázis-szerkesztő táblájan alapuló **Recordset** objektum esetén **True**, azaz az objektumhoz használható könyvjelző. Más adatbázisok, így például egy elsődleges kulccsal nem rendelkező csatolt Paradox táblán alapuló **Recordset** nem támogatja a könyvjelzőket, vagyis ilyen esetben azok nem használhatók.

Egy **Recordset** objektum létrehozásakor vagy megnyitásakor minden egyes rekordnak egyedi, nem a rekord számával azonos értékű könyvjelzője van. Az aktuális rekordhoz tartozó könyvjelzőt a **Bookmark** tulajdonság értékét egy változóba beállítva tárolhatjuk. Ha egy másik rekordra átlépést követően ismét szükségünk van erre a rekordra, a gyors eléréshez elegendő csak a **Recordset** objektum **Bookmark** tulajdonságába ezt a változóértéket beállítani.

A könyvjelzőt nem aktuális rekordhoz úgy állíthatjuk be, hogy a kívánt rekordra lépünk és a **Bookmark** tulajdonság értékét a rekordot azonosító **String** változónak adjuk. A beállítható könyvjelzők száma nincs korlátozva.

Egy **Recordset** objektumból a **Clone** metódussal készített másolat esetén a **Bookmark** tulajdonság beállításai az alap- és a másodpéldány azonosak maradnak és

egymást helyettesítően használhatók. Különböző **Recordset** objektumoknál ez még akkor sem igaz, ha azok ugyanazon objektumból vagy ugyanazzal az SQL utasítással jöttek létre.

A **Bookmark** tulajdonságba törölt rekordhoz tartozó érték beállítása kezelhető hibát okoz.

Egy rekord tartalmának frissítéséhez a **Bookmark** tulajdonságát saját magára kell beállítani. Ez a frissítési módszer érvénytelenít minden előző **Edit** és **AddNew** metódus által hívott műveletet.

A tulajdonság a csak előre gördíthető pillanatfelvétel típusú **Recordset** objektumon nem használható.

Meg kell jegyeznünk, hogy a Visual Basic 3.0-s verziójában a **Bookmark** tulajdonság karakterláncként volt kezelve, azonban a 4.0-s verzió belsőleg **Byte** típusú tömbként tárolja, ezért a **Bookmark** tulajdonság összehasonlítási műveletben történő közvetlen használata kezelhető hibát okoz. Az összehasonlításhoz még a **Bookmark** tulajdonsághoz történő hozzáférés előtt a könyvjelző értékeit **String** változókba kell másolni és az összehasonlítást ezekkel a változókkal lehet elvégezni.

Bookmarkable

Szintaxis: *objektum*.**Bookmarkable**

A **Bookmarkable** tulajdonság **Recordset** (valamint az elavult **Dynaset**, **Snapshot** és **Table**) objektumon érvényes, és a **Recordset** objektum **Bookmark** tulajdonságával beállítható könyvjelzők használatának támogatását jelző értéket ad vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság **Boolean** típusú értéke és annak értelmezése a következő lehet:

True	A **Recordset** támogatja a könyvjelzők használatát.
False	A **Recordset** nem támogatja a könyvjelzők használatát.

A **Bookmarkable** tulajdonság használatával a **Bookmark** tulajdonság beállítása vagy alkalmazása előtt lehet ellenőrizni, hogy az adott **Recordset** támogatja-e a könyvjelző használatát vagy sem. Ez utóbbi esetet a **Bookmarkable** tulajdonság **False** értéke jelzi és a tulajdonság használata kezelhető hibát okoz. A **Bookmarkable** tulajdonság egy eredetileg Microsoft Jet adatbázis-szerkesztő táblájáan alapuló **Recordset** objektum esetén **True**, azaz az objektumhoz — a csak előre gördíthető pillanatfelvétel típusúak kivételével — használható könyvjelző. Más adatbázisok, így például egy elsődleges kulccsal nem rendelkező csatolt Paradox táblán alapuló **Recordset** nem támogatja a könyvjelzőket, vagyis ilyen esetben azok nem használhatók.

BorderColor

Szintaxis: *objektum*.**BorderColor** [= *érték*]

A **BorderColor** tulajdonság **Line** és **Shape** vezérlőelemeken érvényes, és az objektum-határvonal színének kiolvasására vagy beállítására használható.

A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a határvonal színét meghatározó konstans vagy színkód értéke.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
Normál RGB szín	A Color paletta vagy a programkódban az **RGB** vagy **QBColor** függvénnyel meghatározott szín.
Alapértelmezett szín	A Visual Basic objektumtárban a rendszer színkonstansával meghatározott szín. A rendszer alapértelmezett színét a **vbWindowText** konstans értéke határozza meg. A Windows környezet a felhasználó színválasztását a Control Panel érvényes színével helyettesíti.

A normál RGB színeket meghatározó *érték* számértékének érvényes tartománya 0 és 16 777 215 (**&H**FFFFFF) között lehet. A felső byte értéke ebben a tartományban 0, az alsó bitcsoport a legalacsonyabb bittől felfelé a vörös, zöld és kék színösszetevőt határozza meg. Ezen összetevők 0 és 255 (**&H**FF) közötti értékek lehetnek. Ha a felső byte nem nulla, a Visual Basic a Control Panel beállításának és az Object Browser objektumtárában található konstansoknak megfelelő rendszerszíneket használja.

BorderStyle

Szintaxis: *objektum*.**BorderStyle** [= *érték*]

A **BorderStyle** tulajdonság **DBCombo**, **DBList**, **Grid**, **Image**, **Label**, **Line**, **OLE** befogadó, **PictureBox**, **Shape** és **TextBox** vezérlőelemeken, valamint objektumon érvényes, és az objektum keretstílusának kiolvasására vagy beállítására használható. A **TextBox** vezérlőelemen és **Form** objektumon futásidőben csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a keret stílusát meghatározó **Integer** érték.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
0	*None* — Nincs keret vagy azzal kapcsolatos elem.
1	*Fixed Single* — Rögzített méretű keret. Magában foglalja a Control menü mezőjét, a címsort, az ikonállapot és a teljes méret gombokat.
2	*Sizable* — Változtatható méretű, újraméretezhető keret. Magában foglalja a Control menü mezőjét, a címsort, az ikonállapot és a teljes méret gombokat. Alapértelmezett.
3	*Fixed Dialog* — Rögzített méretű, nem újraméretezhető keret. Magában foglalja a Control menü mezőjét és a címsort, de az ikonállapot és a teljes méret gombokat nem.
4	*Fixed ToolWindow* — Rögzített méretű, nem újraméretezhető keret. A 16-bites Windows, és Windows NT 3.51 és előbbi verziókban a *Fixed Single* beállítással azonos tulajdonságú, azonban az ikonállapot és a teljes méret gombokat nem jeleníti meg. A Windows 95 alatt megjeleníti a Close gombot és a csökkentett karakterméretű címsorszöveget. Az űrlap nem jelenik meg a Windows 95 feladatsorában.
5	*Sizable ToolWindow* — Változtatható méretű, újraméretezhető keret. A 16-bites Windows, és Windows NT 3.51 és előbbi verziókban a *Sizable* beállítással azonos tulajdonságú, azonban az ikonállapot és a teljes méret gombokat nem jeleníti meg. A Windows 95 alatt megjeleníti a Close gombot és a csökkentett karakterméretű címsorszöveget. Az űrlap nem jelenik meg a Windows 95 feladatsorában.

A **BorderStyle** tulajdonság beállításai **Grid**, **Image**, **Label**, **OLE** befogadó, **PictureBox** és **TextBox** vezérlőelemeken a következők lehetnek:

Ha az *érték*:	Értelmezése:
0	*None* — Nincs keret vagy azzal kapcsolatos elem. Az **Image** és **Label** vezérlőelemeken alapértelmezett.
1	*Fixed Single* — Rögzített méretű keret. A **Grid**, **PictureBox**, **TextBox** és **OLE** befogadó vezérlőelemen alapértelmezett.

A **BorderStyle** tulajdonság beállításai **Line** és **Shape** vezérlőelemeken a következők lehetnek:

Ha az *érték*:	Értelmezése:
0	*Transparent* — Átlátszó.
1	*Solid* — Folytonos. A keret az alakzat éléhez központosított. Alapértelmezett.
2	*Dash* — Szaggatott vonal (- — -).
3	*Dot* — Pontozott vonal (....).
4	*Dash-dot* — Vonal-pont (-.-.-).
5	*Dash-dot-dot* — Vonal-pont-pont (..-..-..-).

6 *Inside solid* — Belül folytonos. A keret külső éle az alakzat külső élével azonos.

A **BorderStyle** tulajdonság űrlapok esetén jelentősen meghatározza azok vizuális megjelenését, általános ablakkénti vagy párbeszédpanelkénti megkülönböztethetőségét. A standard párbeszédpanelek esetén általában a **BorderStyle** = 3 (*Fixed Dialog*) beállítás, míg a **BorderStyle** = 4 (*Fixed ToolWindow*) és **BorderStyle** = *5 (Sizable ToolWindow*) beállítás az eszköztár stílusú ablakok létrehozásához használatos.

A **BorderStyle** = 2 (*Sizable*) beállításnál futásidőben az MDI-alűrlapok a Windows operációs környezet által alapértelmezett méretben jelennek meg az MDI-űrlapon belül. Minden más beállításnál az űrlap a programfejlesztési fázisban megadott méretben jelenik meg.

Egy **Form** objektum **BorderStyle** tulajdonságának módosítása egyben a **MinButton**, **MaxButton** és **ShowInToolbar** tulajdonságokat is módosítja:

- Ha a **BorderStyle** = 1 (*Fixed Single*), a **MinButton**, **MaxButton** és **ShowInToolbar** automatikusan **True** értékű lesz.
- Ha a **BorderStyle** = 0 (*None*), 3 (*Fixed Dialog*), 4 (*Fixed ToolWindow*) vagy 5 (*Sizable ToolWindow*), a **MinButton**, **MaxButton** és **ShowInToolbar** automatikusan **False** értékűvé válik.

A menüvel rendelkező űrlap **BorderStyle** = 3 (*Fixed Dialog*) beállítása esetén az űrlap az 1 (*Fixed Single*) beállításnak megfelelően jelenik meg.

A megjelenítendő űrlap modalitását futásidőben a **Show** metódussal lehet beállítani.

BorderWidth

Szintaxis: *objektum*.**BorderWidth** [= *érték*]

A **BorderWidth** tulajdonság **Line** és **Shape** vezérlőelemeken érvényes, és az objektum keretszélességének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a keret szélességét meghatározó 1 és 8192 közötti **Integer** érték.

A **BorderWidth** és a **BorderStyle** tulajdonságok együttesen határozzák meg a vezérlőelem keretének megjelenését. A **BorderStyle** értéke a következőképpen befolyásolja a **BorderWidth** tulajdonságot:

Ha a **BorderStyle** értéke:	Hatása a **BorderWidth** tulajdonságra:
0	A **BorderWidth** beállítás hatástalan.
1-5	A keretvonal szélessége a középpont felől kifelé terjed, a vezérlőelem magassága és szélessége a keret közepétől számított.
6	A keretvonal szélessége a kerettől befelé terjed, a vezérlőelem magassága és szélessége a keret külsejétől számított.

Ha a **BorderWidth** értéke nagyobb mint 1, a **BorderStyle** tulajdonság értéke csak 1 (*Solid*) vagy 6 (*Inside Solid*) lehet.

BoundColumn

Szintaxis: *objektum*.**BoundColumn** [= *érték*]

A **BoundColumn** tulajdonság **DBCombo** és **DBList** vezérlőelemeken érvényes, és egy **Recordset** objektumban egy másik **Recordset** objektumnak adatértéket adó forrásmező nevének, illetve egy **DBCombo** vagy **DBList** vezérlőelem feltöltésére használandó mező nevének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a **RowSource** tulajdonsággal adott **Data** vezérlőelemmel létrehozott **Recordset** egy mezőjének nevét megadó karakteres kifejezés.

Általában a **DBCombo** és **DBList** vezérlőelemekkel dolgozva két **Data** vezérlőelemet használunk: az elsőt a **RowSource** tulajdonsággal kijelölt listának a **ListField** tulajdonsággal kijelölt mezővel való feltöltéséhez, a második — a **DataSource** tulajdonsággal kijelölt — **Data** vezérlőelemet pedig a frissítendő mezőt tartalmazó **Recordset** objektum kezeléséhez. Amikor a lista egy elemét a felhasználó kiválasztja, az adat a **RowSource** rekordhalmazának **BoundColumn** mezőjéből a második, a **DataSource** és **DataField** tulajdonságokkal kijelölt **Data** mezőjébe kerül. Ilyen módon ugyanazon **Recordset** egy mezőjét a lista feltöltéséhez, egy másikat pedig a **DataSource** és **DataField** tulajdonságokkal kijelölt **Recordset** objektumnak lehet átadni.

Ha a **ListField** tulajdonság nincs megadva, a vezérlőelem listájának feltöltése a **BoundColumn** tulajdonsággal meghatározott mezővel történik. Ha a **Recordset** objektumban nem található a **BoundColumn** tulajdonsággal meghatározott mező, kezelhető hiba következik be.

A lista feltöltéséhez és a frissítendő mezőt tartalmazó **Recordset** objektum kezeléséhez egy-egy **Data** vezérlőelemet használva a frissített rekord értékeinek nem szükségszerűen megfelelő értékek listája hozható létre.

BoundText

Szintaxis: *objektum*.**BoundText** [= *érték*]

A **BoundText** tulajdonság **DBCombo** és **DBList** vezérlőelemeken érvényes, és a vezérlőelem **BoundColumn** tulajdonságának egy kiválasztás után a **DataField** tulajdonságnak átadandó vagy abból átveendő értékének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, az adatértéket megadó karakteres kifejezés.

A **DBCombo** és **DBList** vezérlőelemekkel történt kiválasztás után a **BoundText** tulajdonság a **BoundColumn** mező tartalmát fogja tartalmazni. A vezérlőelem listarésze jellemzően egy adatbázis-lekérdezés alapján épül fel. A lista egy elemének kiválasztásakor az adott elem a **DataSource** tulajdonsággal adott **Data** vezérlőelem által használhatóvá válik. A kiválasztás a vezérlőelem szövegbeviteli mezőjében is megjelenik, ahol szerkeszthető lesz. Ha a felhasználó a szövegbeviteli mezőbe egy értéket ír, a listarész megkísérli az annak megfelelő pozícionálást. Egyezésnél a **BoundText** tulajdonság értéke a **BoundControl** tulajdonsággal adott mező alapján állítódik be, egyezés hiányában pedig **Null** értéket kap.

CacheSize

Szintaxis: *objektum*.**CacheSize** [= *érték*]

A **CacheSize** tulajdonság eredményhalmaz típusú (csak ODBC) **Recordset** objektumon érvényes, és az objektumnak egy ODBC adatforrásból helyileg tárolt rekordjainak számát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a rekordok számát 5 és 1200 között (de legfeljebb a rendelkezésre álló memória méretéig) megadó **Long** típusú numerikus kifejezés. Általában a 100 érték megfelelő. Ha az *érték* = 0, a rekordok átmeneti tárolása kikapcsolódik.

Adatok átmeneti gyorsítótárban történő tárolása jelentősen javítja a távoli kiszolgálótól eredményhalmaz típusú **Recordset** objektumokon keresztül adatot kiolvasó alkalmazás jellemzőit. Az átmeneti gyorsítótár a helyi memória egy területe, amelyen a kiszolgálótól az alkalmazás futása alatt adatot kérő eseményben leggyakrabban szükséges adatok helyezhetők el. Ha adatra van szükség, a Microsoft Jet először a gyorsítótárban keresi a több időt igénylő kiszolgálótól történő adatkérés helyett. A Microsoft Jet a

gyorsítótárból annak tárolási tartományán belül kér adatokat, az azon kívüli rekordokért pedig a kiszolgálókhoz fordul. A nem ODBC adatforrás által tartalmazott adatok nem kerülnek az átmeneti gyorsítótárba.

Minden ODBC adatforrásnak — így egy csatolt táblának — lehet helyi gyorsítótára. A gyorsítótár létrehozásához a távoli adatforrásból meg kell nyitni a **Recordset** objektumot, be kell állítani a **CacheSize** és **CacheStart** tulajdonságokat, majd vagy a **FillCache** metódust kell használni, vagy a **Move** metódusokkal a rekordokon végig kell lépkedni.

A **CacheSize** tulajdonság értéke az alkalmazás által egy időben használható rekordok száma lehet.

Természetesen az átmeneti gyorsítótárban lévő rekordok nem tükrözik a forrás adataiban más felhasználók által időközben végrehajtott módosításokat. Az átmeneti tár adatainak frissítéséhez a **Recordset** objektum **CacheSize** tulajdonságának értékét nullára kell beállítani, majd a tárméretet az eredetire kell visszaállítani és ezt követően a **FillCache** metódust kell használni.

CacheStart

Szintaxis: *objektum*.**CacheStart** [= *érték*]

A **CacheStart** tulajdonság eredményhalmaz típusú **Recordset** objektumon érvényes, és az objektumban egy ODBC adatforrás helyileg tárolt első rekordjához tartozó könyvjelző kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a rekord könyvjelzőjét megadó **String** típusú karakteres kifejezés.

Az adatok átmeneti gyorsítótárban történő tárolása jelentősen javítja a távoli kiszolgálótól eredményhalmaz típusú **Recordset** objektumokon keresztül adatot kiolvasó alkalmazás jellemzőit. Az átmeneti gyorsítótár a helyi memória egy területe, amelyen a kiszolgálótól az alkalmazás futása alatt adatot kérő eseményben legutoljára kért adatok helyezhetők el. Amikor adatra van szükség, a Microsoft Jet először a gyorsítótár tartalmát ellenőrzi a több időt igénylő kiszolgálótól történő adatkérés helyett. A nem ODBC adatforrás által tartalmazott adatok nem kerülnek az átmeneti gyorsítótárba.

A **CacheStart** tulajdonság az adott átmeneti gyorsítótárhoz tartozó **Recordset** objektum első rekordjának könyvjelzője. Bármelyik rekord könyvjelzője használható a **CacheStart** tulajdonság beállításához. Az átmenetileg tárolt rekordok közül elsőként használandót az aktuális rekorddá tételével, majd a könyvjelzőjét a **CacheStart** tulajdonság értékeként beállítva jelölhetjük ki.

Cancel

Szintaxis: *objektum*.**Cancel** [= *érték*]

A tulajdonság **CommandButton** és **OLE** befogadó vezérlőelemeken érvényes, és egy űrlapon a parancsgomb Cancel gombkénti használatára jellemző érték beállítására vagy kiolvasására alkalmas. A parancsgomb **CommandButton** vezérlőelem vagy egy **OLE** befogadó vezérlőelemen belül bármely, parancsgombként funkcionáló objektum lehet. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

A tulajdonság **Boolean** típusú értéke és annak értelmezése a következő lehet:

True A **CommandButton** vezérlőelem Cancel gomb.
False A **CommandButton** vezérlőelem nem Cancel gomb. (Alapértelmezett.)

A **Cancel** tulajdonság a felhasználó számára az előzőleg bekövetkezett változások érvénytelenítését, az űrlap előző állapotának visszaállítását teszi lehetővé.

Egy űrlapon csak egyetlen **CommandButton** vezérlőelem **Cancel** tulajdonsága lehet **True**, az összes többi parancsgomb ugyanezen tulajdonsága automatikusan **False** értékűvé válik. Ha egy aktív űrlap **CommandButton** vezérlőelemének **Cancel** tulajdonsága **True**, a felhasználó ezt a parancsgombot rákattintással, az ESC vagy — fókuszban lévő parancsgomb esetén — az ENTER billentyű leütésével is működtetheti.

Az **OLE** befogadó vezérlőelem esetén a **Cancel** tulajdonság csak azon objektumoknál használható, amelyek parancsgombként funkcionálnak.

A tulajdonságot — az előző változatokból ismert módon — a végzetes, visszaállíthatatlan hatású, például fájlokat törlő műveleti parancsot hatástalanító gombon használva, a műveletet csak szándékosan, esetleg egy figyelmeztetés megjelenítése után tehetjük csak végrehajthatóvá. Ehhez a **Cancel** és a **Default** tulajdonságot is **True** értékre kell beállítani.

CancelError

Szintaxis: *objektum*.**CancelError** [= *érték*]

A tulajdonság **CommonDialog** vezérlőelemeken érvényes, és a Cancel gomb kiválasztáskor hiba bekövetkezését jelző érték beállítására vagy kiolvasására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a hiba bekövet-

kezését jelző **Boolean** típusú kifejezés.

A tulajdonság **Boolean** típusú értéke és annak értelmezése a következő lehet:

True A parancsgomb használatakor hiba következik be.
False A parancsgomb használatakor nem következik be hiba. (Alapértelmezett.)

A tulajdonság **True** értéke esetén a Cancel gomb minden egyes kiválasztása 32755 (**cdlCancel**) számú hibát okoz.

Caption

Szintaxis: *objektum*.**Caption** [= *érték*]

A tulajdonság **CheckBox**, **CommandButton**, **Data**, **Frame**, **Label**, **Menu** és **OptionButton** vezérlőelemeken, valamint **Form**, **MDIForm** és **MenuLine** objektumon érvényes, és az objektumoktól függően eltérő hatású: **Form** és **MDIForm** objektumon az objektum címsorában, illetve ikonméretű űrlap esetén az ikon alatt megjelenő szöveget, vezérlőelemek esetén a vezérlőelemben vagy az mellett megjelenő szöveget, **MenuLine** objektum esetén pedig a **Menu** vezérlőelem vagy a **MenuItems** halmazbeli objektum által megjelenített szöveget határozza meg. A **Menu** vezérlőelem esetén a **Caption** tulajdonság általában írható és olvasható is, azonban az olyan menüknél csak olvasható, amelyeket a Visual Basic jelenít meg vagy ad át, mint például az Add-Ins menü Add-In Manager parancsa.

A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az *objektum* az aktív űrlapmodulhoz kapcsolódó űrlap lesz.
- Az *érték* argumentum a címsorban megjelenítendő szöveget tartalmazó karakteres kifejezés.

Az új objektum létrehozásakor annak alapértelmezett címfelirata a **Name** tulajdonság alapértelmezett értéke lesz. Az alapértelmezett felirat az objektum nevét és egy sorszámot (például Form1 vagy Command1) tartalmaz. Ha ennél informatívabb címre van szükség, ezt a **Caption** tulajdonságban szereplő karakteres kifejezésként adhatjuk meg.

A **Caption** tulajdonságot egy vezérlőelemhez közvetlen hozzáférést adó hívóbetű kijelölésére is használhatjuk, közvetlenül a kívánt karakter elé helyezett & karakterrel. A választott karakter ezt követően aláhúzottan jelenik meg, jelezve, hogy a program futásakor a vezérlőelem a billentyűzetről az ALT és az aláhúzott karakternek megfelelő billentyű egyidejű leütésével is fókuszba hozható. (A címsorban a & nem hívóbetűkénti használata && alakban történhet.)

A karakteres kifejezés hossza **Label** vezérlőelem esetén 2048, űrlapoknál és egyéb vezérlőelemeknél 255 karakternyire korlátozott.

Az űrlap címszövegének megjelenítéséhez a **BorderStyle** tulajdonság 1 (**vbFixedSingle**), 2 (**vbSizable**) vagy 3 (**vbFixedDouble**) beállításai használhatók.

A címfelirat szövegének az űrlap címsávjánál hosszabb része elvész, kivéve a címkéket, ahol **True** értékű **AutoSize** tulajdonságot beállítva a címke nagysága követi a címfelirat hosszát.

CellSelected

Szintaxis: *objektum*.**CellSelected**

A **Grid** vezérlőelemen használható tulajdonsággal az olvasható ki, hogy a **Col** és **Row** tulajdonságokkal kijelölt aktív cella a vezérlőelem kijelölt cellája-e vagy a kijelölt tartományán belül van-e. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság **Boolean** típusú értéke és annak értelmezése a következő lehet:

True A meghatározott cella a választott tartományon belül van.
False A meghatározott cella a választott tartományon kívül van.

Futásidőben a felhasználó a cellatartományt az egérrel a tartomány első cellára kattintva, majd lenyomott egérgombbal a tartomány végéig mozogva, illetve a `SHIFT` billentyűt és a mutatónyilak egyidejű használatával jelölheti ki.

A kijelölt cella meghatározása a **Col** és **Row**, a választott tartomány meghatározása pedig a **SelStartCol**, **SelEndCol**, **SelStartRow** és **SelEndRow** tulajdonságokkal lehetséges.

Checked

Szintaxis: *objektum*.**Checked** [= *érték*]

A tulajdonság **Menu** vezérlőelemen és **MenuLine** objektumon érvényes, és a menüelem melletti "pipa" megjelenését meghatározó érték kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum a "pipa" megjelenítését meghatározó **Boolean** típusú kifejezés.

A tulajdonság **Boolean** típusú értéke és annak értelmezése a következő lehet:

True A választott tételt a mellette megjelenő kipipálás jelzi.
False A választott tételt nem jelzi kipipálás. (Alapértelmezett.)

A programfejlesztési fázisban a Menu Editor használható a **Checked** tulajdonság

True értékre állításához. A futásidőben egy **Menu** vezérlőelemhez kapcsolt **Click** eseményeljárásban kapcsolhatjuk át a **Checked** tulajdonság értékét.

Futásidőben a tulajdonság egy menü-vezérlőelemhez tartozó **Click** eseményeljárásban, az űrlapok betöltésére használt **Load**, illetve a bejelentkező képernyőt betöltő automatikus programindítási eljárásban használatos.

A **Menu** vezérlőelem esetén a **Checked** tulajdonság rendes körülmények között írható és olvasható is, azonban az olyan menüknél csak olvasható, amelyeket a Visual Basic jelenít meg vagy ad át, mint például az Add-Ins menü Add-In Manager parancsa.

Class

Szintaxis: *objektum*.**Class**[= *érték*]

A tulajdonság **OLE** befogadó vezérlőelemen érvényes, és egy beágyazott objektum osztálynevének kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum az osztálynevet tartalmazó karakteres kifejezés.

Az osztálynév egy objektum típusát határozza meg. Az OLE vagy OLE Automatizmust támogató alkalmazások az objektumaik osztálynevét az *alkalmazás.objektumtípus.verzió* vagy *objektumtípus.verzió* alakban adják meg. Az osztálynév szintaxisában használt elemek értelmezése a következő:

Az osztálynév rész:	Értelmezése:
alkalmazás	Az objektumot szolgáltató alkalmazás neve.
objektumtípus	Az objektumtár szerinti objektumnév.
verzió	Az objektum vagy az objektumot szolgáltató alkalmazás verziószáma.

Ennek megfelelően például a Microsoft Excel 5.0 számos objektumot, így munkalapokat és diagramokat is támogat, amelyek osztályneve **Excel.Sheet.5** vagy **Excel.Chart.5** alakú lehet, míg a Microsoft WordArt 2.0 csak egyetlen objektumot támogat **MSWordArt.2** osztálynévvel. (Néhány OLE programozási dokumentáció az osztálynév szintaxisára programozási azonosítóként hivatkozik.)

A programfejlesztési fázisban egy objektumot az Insert Object és Paste Special párbeszédpanel használatával létrehozva, az **OLE** befogadó vezérlőelem **Class** tulajdonságának beállítása automatikusan történik a Properties ablakban. A használható osztálynevek listáját a Properties ablakban a Class tulajdonságot kijelölve, majd a Properties gombra kattintva tekinthetjük meg.

A **Class** tulajdonság az objektumnak a rendszer vágólapjáról történő átmásolásával frissítődik, például egy Microsoft Excel diagramot a vágólapról egy előzőleg Excel munkalapot tartalmazó **OLE** befogadó vezérlőelembe másolva a **Class** tulajdonság az **Ex-**

cel.Sheet.5 értékről **Excel.Chart.5** értékre változik. A másolás futásidőben a **Paste** vagy a **PasteSpecialDlg** metódussal történhet.

Clip

Szintaxis: *objektum*.**Clip**[= *érték*]

A programfejlesztési fázisban nem használható tulajdonság **Grid** vezérlőelemen érvényes, és vezérlőelem kiválasztott tartományában a cellatartalom kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum a cella tartalmát tartalmazó karakteres kifejezés.

Az *érték* argumentum több oszlop és sor tartalmát is magában foglalhatja, ahol a karakteres kifejezésben a sor egy új celláját a TAB karakter (ANSI kódja 9), egy új sort pedig a kocsivissza karakter (ANSI kódja 13) jelöli. Ezeket a taglókaraktereket a karakteres kifejezések szerkesztésénél a **Chr( )** függvénnyel lehet beépíteni.

Az adatoknak egy **Grid** vezérlőelembe helyezésekor csak a kijelölt cellák módosulnak. Ha az *érték* karakteres kifejezésében kevesebb elem van, mint a kijelölt tartományban, a fennmaradó cellák **Null** értéket kapnak, míg a tartomány elemeinél több adatot tartalmazó kifejezésből a szükségtelen rész figyelmen kívül marad.

ClipControls

Szintaxis: *objektum*.**ClipControls** [= *érték*]

A tulajdonság **Form** objektumon, valamint **Frame** és **PictureBox** vezérlőelemen érvényes, és a **Paint** események által a teljes objektum vagy csak az újonnan megjelenő területek újrarajzolását meghatározó érték kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum az újrarajzolás módját meghatározó **Boolean** típusú kifejezés.

A tulajdonság **Boolean** típusú értéke és annak értelmezése a következő lehet:

True A **Paint** esemény grafikus metódusai a teljes objektumot újrarajzolják. A **Paint** esemény előtt az űrlapon a nem grafikus objektumokat körülvevő kizárási tartomány jön létre. (Alapértelmezett.)

False A **Paint** esemény grafikus metódusai csak az újonnan megjelent területet rajzolják újra, és nem jön létre kizárási tartomány az űrlapon a nem grafikus objektumokat körül. Ennél az *érték* beállításnál az összetett alakzatok rendszerint gyorsabban tölthetők be és rajzolhatók újra.

A kizárás az az eljárás, amely meghatározza, hogy például egy **Form** objektum vagy **PictureBox** vezérlőelem mely részét kell az űrlap megjelenítéskor megrajzolni. Az űrlap és a vezérlőelemek körvonala a memóriában jön létre, és a Windows operációs környezet ezen körvonalakat használja néhány rész, így például a háttér oly módon történő megrajzolására, hogy az ne hasson más részekre, például egy **TextBox** vezérlőelem tartalmára. Mivel a kizárási tartomány a memóriában jön létre, a **ClipControls** tulajdonság **False** értékénél, a rajzolás vagy újrarajzolás kevesebb időt igényel, a működés felgyorsul.

A kizárt tartomány az **Image**, **Label**, **Line** és **Shape** vezérlőelemet kivéve magában foglalja a többi vezérlőelemet.

A vezérlőelemek egymásba ágyazása helytelen újrarajzolást eredményezhet, ha a **ClipControl** tulajdonság **False** beállításával rendelkező vezérlőelemen belül egy olyan vezérlőelemet használunk, amelynek **ClipControl** tulajdonsága **True**. Az ilyen problémák megelőzésére mindkét (a befogadó és a beágyazott) vezérlőelem **ClipControl** tulajdonságának **True** értékűnek kell lennie.

Clustered

Szintaxis: *objektum*.**Clustered** [= *érték*]

A tulajdonság csak nem Jet adatbázis **Index** objektumán érvényes, és egy olyan érték kiolvasására vagy beállítására alkalmas, amely azt jelzi, hogy az objektum tábla csoportosítási indexe-e. A tulajdonság írható és olvasható az olyan új **Index** objektumon, amely még nem lett egy halmazhoz hozzáadva, és csak olvasható az **Indexes** halmaz egy létező **Index** objektumán. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum a csoportosítási indexet meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges beállítása és annak értelmezése a következő lehet:

True Az **Index** egy csoportosítási indexet jelöl.
False Az **Index** nem csoportosítási indexet jelöl.

A csoportosítási index egy vagy több nem-kulcsmezőt tartalmazhat, együttesen, és minden rekordot a táblán előre megadott módon rendezve. A csoportosítási index hatékony hozzáférést biztosít egy táblán belül a nem egyedi indexértékű rekordokhoz.

A tulajdonság használatával kapcsolatban megjegyzendő, hogy a Microsoft Jet adat-

bázis-szerkesztőt használó adatbázisok figyelmen kívül hagyják az **Index** tulajdonság értékét, mivel a Jet nem támogatja a csoportosítási indexeket. A tulajdonság ODBC adatforráshoz sem használható.

A táblákhoz nem kötelező indexeket létrehozni, azonban másként a nagyméretű, nem indexelt tábla adott rekordjához történő hozzáférés sok időt igényel. Az **Index** objektumban minden egyes **Field** objektum **Attributes** tulajdonsága meghatározza a rekordok sorrendjét, és ebből következően az index elérési módját.

Col

Szintaxis: *objektum*.**Col** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **DBGrid** és **Grid** vezérlőelemen érvényes, és a vezérlőelem aktív cellájának kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum az aktív cellát tartalmazó oszlop számát megadó **Integer** érték.

A tulajdonságot a **Grid** vezérlőelemben az aktív cella kijelölésére vagy annak meghatározására használhatjuk, hogy az aktív cella melyik oszlopban található. Az oszlopok számozása nullától, azaz a bal oldali első oszloptól kezdődik. A tulajdonság futásidejű beállítása nem módosítja, hogy mely cellák kiválasztottak. Egy cellatartomány beállítására a **SelStartCol**, **SelEndCol**, **SelStartRow** és **SelEndRow** tulajdonságokat lehet használni.

ColAlignment

Szintaxis: *objektum*.**ColAligment**(*oszlop*) [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Grid** vezérlőelemen érvényes, és egy oszlopban az adat helyzetét megadó érték kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *oszlop* argumentum egy oszlop száma.
- Az *érték* argumentum az adat helyzetét meghatározó **Integer** érték.

Az **Integer** típusú *érték* lehetséges beállításai és azok értelmezése a következő:

Ha az *érték*:	Értelmezése:
0	Az adatok balra igazítottak.(Alapértelmezett.)
1	Az adatok jobbra igazítottak.
2	Az adatok középre igazítottak.

Az adattáblázat minden egyes oszlopa egyedileg beállítható **ColAlignment** tulajdonságú lehet. A **ColAlignment** tulajdonság értéke nem befolyásolja a rögzített oszlopok celláit, azokban a helyzetet a **FixedAlignment** tulajdonsággal lehet beállítani.

ColIndex

Szintaxis: *objektum*.**ColIndex** [= *érték*]

Ez a tulajdonság **Column** objektumon érvényes, és a **DBGrid** vezérlőelem **Columns** halmazában egy oszlop helyzetére és a látható pozíciójára jellemző érték kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum a bal szélső oszlopot jelölő egy 0 és a **Count** tulajdonság értéke közötti **Integer** érték.

Az oszlopokat a **ColIndex** tulajdonság értékének módosításával lehet átrendezni. A tulajdonság 0 értéke esetén az oszlop a bal szélső pozícióba kerül és az összes többi, az oszlop eredeti pozíciójától balra lévő oszlop **ColIndex** értéke megnő.

A **ColIndex** tulajdonságot a **DBGrid** vezérlőelem **Columns** halmazában a **Count** tulajdonság értékének megfelelően beállítva az oszlop a jobb oldali legszélső pozícióba kerül és az összes többi, eredetileg az oszloptól jobbra lévő oszlop **ColIndex** értéke csökken.

A **ColIndex** tulajdonságot 0 és a **DBGrid** vezérlőelem **Columns** halmazában a **Count** tulajdonság értéke közötti *n* számra beállítva az oszlop a bal oldalról számított *n*-edik oszlop bal oldalára kerül és az összes többi, az eredeti pozíció és *n* közötti oszlop **ColIndex** értéke ennek megfelelően módosul.

ColIsVisible

Szintaxis: *objektum*.**ColIsVisible**(*index*)

Ez a tulajdonság **Grid** vezérlőelemen érvényes, és a vezérlőelem adott oszlopának aktuális láthatóságát jellemző érték kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum a vezérlőelembeli oszlopot megadó, 0 és a **Cols** tulajdon-

ság értéke -1 közötti **Integer** érték.

A **ColIsVisible** tulajdonság visszaadott értékei és azok értelmezése a következő:

True	Az adott oszlop aktuálisan látható.
False	Az adott oszlop aktuálisan nem látható.

CollatingOrder

Szintaxis: *objektum*.**CollatingOrder**

A **Database** és **Field** objektumokon érvényes, de csak olvasható tulajdonság a szövegek összehasonlítási módját meghatározó jellemző értéket ad vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **CollatingOrder** tulajdonság visszaadott értékei és azok értelmezése a következő:

Ha a szimbolikus konstans:	Az összehasonlítási mód:
dbSortGeneral	Az angol, német, francia, portugál, olasz és modern spanyol nyelvnek megfelelően.
dbSortArabic	Az arab nyelvnek megfelelően.
dbSortCyrillic	Az orosz nyelvnek megfelelően.
dbSortCzech	A cseh nyelvnek megfelelően.
dbSortDutch	A holland nyelvnek megfelelően.
dbSortGreek	A görög nyelvnek megfelelően.
dbSortHebrew	A héber nyelvnek megfelelően.
dbSortHungarian	A magyar nyelvnek megfelelően.
dbSortIcelandic	Az izlandi nyelvnek megfelelően.
dbSortJapanese	A japán nyelvnek megfelelően.
dbSortNeutral	Semleges összehasonlítási mód.
dbSortNorwdan	A norvég vagy dán nyelvnek megfelelően.
dbSortPDXIntl	A Paradox nemzetközi összehasonlítási módja szerint.
dbSortPDXNor	A Paradox norvég vagy dán összehasonlítási módja szerint.
dbSortPDXSwe	A Paradox svéd vagy finn összehasonlítási módja szerint.
dbSortPolish	A lengyel nyelvnek megfelelően.
dbSortSpanish	A hagyományos spanyol nyelvnek megfelelően.
dbSortSwedfin	A svéd vagy finn nyelvnek megfelelően.
dbSortTurkish	A török nyelvnek megfelelően.
dbSortUndefinied	Az összehasonlítási mód határozatlan vagy ismeretlen.

A konstansokat a DAO (Data Access Object) objektumtár tartalmazza.

Field objektumnál a **CollatingOrder** tulajdonság használata attól a **Fields** halmazt tartalmazó objektumtól függ, amelyhez a **Field** tartozik:

Ha az objektum:	A tulajdonság használata:
Index	Nem támogatott.
QueryDef	Csak olvasható.
Recordset	Csak olvasható.
Relation	Nem támogatott.
TableDef	Csak olvasható.

A **CollatingOrder** tulajdonság értéke az adatbázis létrehozásakor a **CreateDatabase** metódus, illetve a legutolsó tömörítéskor a **CompactDatabase** metódus *hely* argumentumában használt értéknek felel meg.

Az adatbázis vagy mező karakteres összehasonlítási módjának meghatározása egy **Database** vagy **Field** objektum **CollatingOrder** tulajdonságának vizsgálatával lehetséges. A **Field** objektumot az azt tartalmazó **Database** objektumtól eltérőként az új **Field** objektum **CollatingOrder** tulajdonságával lehet beállítani.

Valamely indexben az összehasonlítási módot és irányt az **Index** objektum **Fields** halmazában lévő **Field** objektum **CollatingOrder** és **Attributes** tulajdonságai együttesen szabják meg. Az összehasonlítási mód csak egy teljes táblára állítható be, egyedi indexre nem.

A **CollatingOrder** tulajdonság értékét a **StrComp( )** és **InStr( )** függvények feltételes ***compare*** argumentumaként lehet használni.

Color

Szintaxis: *objektum*.**Color** [= *érték*]

Ez a tulajdonság **CommonDialog** vezérlőelemen érvényes, és a kiválasztott színt meghatározó érték kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum a színt meghatározó konstans vagy színkód értéke.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
Normál RGB szín	A programkódban az **RGB** vagy **QBColor** függvénnyel meghatározott szín.
Alapértelmezett szín	A Visual Basic objektumtárban a rendszer színkonstansával meghatározott szín. A Windows környezet a felhasználó színválasztását a Control Panel érvényes színével helyettesíti.

A **Long** típusú RGB színkódnak a Color párbeszédpanelen való kiolvasásához a **RGBInit**, a Font párbeszédpanelen való kiolvasásához pedig az **Effects** jelzőnek beállítottnak kell lennie.

ColorMode

Szintaxis: *objektum*.**ColorMode** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Printer** objektumon érvényes, és a színes nyomtató színes vagy monokróm módban történő működését meghatározó érték kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum a nyomtatási módot meghatározó szimbolikus konstans vagy **Integer** érték.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	A nyomtatás:
vbPRCMMonochrome	1	A nyomtatás monokróm (általában fekete-fehér árnyalatos).
vbPRCMColor	2	Színes nyomtatás.

Az alapértelmezett nyomtatási mód a nyomtató vezérlőprogramtól és az aktuális nyomtatóbeállítástól függ. A nyomtató gyártója által adott vezérlőprogram esetleg egyes tulajdonságok hatástalanságát vagy több tulajdonság azonos hatását eredményezheti. Erre vonatkozóan a nyomtató leírása tartalmazhat további információt.

A tulajdonság értéke a monokróm nyomtatókra hatástalan.

ColPos

Szintaxis: *objektum*.**ColPos**(*index*)

Ez a tulajdonság **Grid** vezérlőelemen érvényes, és a vezérlőelem bal felső sarka és egy adott oszlop bal felső sarka közötti távolság kiolvasására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum a vezérlőelembeli oszlopot megadó, 0 és a **Cols** tulajdonság értéke -1 közötti **Integer** érték.

A **ColPos** tulajdonság visszaadott értéke a távolságot twip egységben adja meg. A twip egy pont huszad része, így 1 (logikai) cm = 567 twip.

Cols

Szintaxis: *objektum*.**Cols** [= *érték*]

Ez a tulajdonság **Grid** vezérlőelemen érvényes, és használatával a vezérlőelem ösz-

szes oszlopának számát lehet kiolvasni vagy beállítani. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum az összes oszlop 1 és 400 között megengedett száma.

A tulajdonsággal futásidőben dinamikusan megnövelhető a **Grid** vezérlőelem minimálisan egy oszlopból és egy sorból álló mérete.

ColumnHeaders

Szintaxis: *objektum*.**ColumnHeaders** [= *érték*]

Ez a tulajdonság **DBGrid** vezérlőelemen érvényes, és használatával a vezérlőelemben az oszlop fejlécének megjelenítésére jellemző értéket lehet kiolvasni vagy beállítani. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum az oszlop fejlécének megjelenítésére jellemző **Boolean** típusú érték.

Az *érték* argumentum értéke és értelmezése a következő lehet:

True	A **DBGrid** vezérlőelem oszlopfejléce megjelenített.
False	A **DBGrid** vezérlőelem oszlopfejléce nem megjelenített.

Columns

Szintaxis 1: *objektum*.**Columns**

Szintaxis 2: *objektum*.**Columns** [= *érték*]

Ezt a tulajdonságot **DBGrid** és **ListBox** vezérlőelemen eltérő szintaxissal és értelmezéssel lehet használni.

A **DBGrid** vezérlőelemen a Szintaxis 1 érvényes, és ekkor a tulajdonság a **Column** objektumok egy halmazát adja vissza. A **Columns** tulajdonság **String** vagy **Variant** típusú lehet. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Column** objektumok tulajdonságainak módosításával a **DBGrid** vezérlőelem legtöbb attribútuma manipulálható. A szükséges **Column** objektumot a **Col** tulajdonság beállításával lehet kijelölni.

A **ListBox** vezérlőelemen a Szintaxis 2 érvényes, és ekkor a tulajdonság a vezérlőelem vízszintes vagy függőleges gördíthetőségét jellemző érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum a gördíthetőséget és az elemek oszlopbeli elrendezését megadó **Integer** érték.

Az *érték* argumentum értéke és értelmezése a következő:

Ha az *érték*:	Értelmezése:
0	Az elemek egyetlen oszlopba rendezettek és a **ListBox** függőlegesen gördíthető. (Alapértelmezett.)
1 — *n*	Az elemek folytatólagosan több oszlopba tördeltek, először az első oszlopot, majd a másodikat és így tovább feltöltve. A **ListBox** vízszintesen gördíthető és *n* számú oszlopot jelenít meg.

A vízszintesen gördíthető **ListBox** vezérlőelemeknél az oszlopszélesség megegyezik a **ListBox** szélességének az oszlopszámnak megfelelő felosztásával kapott értékkel.

A **Columns** tulajdonság futásidőben nem állítható nullára (többoszloposról egyoszloposra) vagy nulláról más értékre (egyoszloposról többoszloposra), de a többoszlopos **ListBox** oszlopainak száma módosítható.

ColWidth

Szintaxis: *objektum*.**ColWidth**(*oszlop*) [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Grid** vezérlőelemen érvényes, és egy adott oszlopban szélességének kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *oszlop* argumentum egy oszlop száma.
- Az *érték* argumentum az oszlop szélességét megadó numerikus kifejezés. A szélesség twip egységben értelmezett. A twip egy pont huszad része, így 1 (logikai) cm = 567 twip.

A **ColWidth** tulajdonsággal futásidőben lehet adott oszlop szélességét beállítani. Megjegyzendő, hogy a felhasználó az oszlopok közötti választóvonal egérrel történő elhúzásával is megváltoztathatja az oszlopszélességet.

Comments

Szintaxis: *objektum*.**Comments**

Ezt a futásidőben csak olvasható tulajdonság az **App** objektumon használható a futó alkalmazásra vonatkozó megjegyzést tartalmazó karakteres kifejezés kiolvasására. A szintaxisban az *objektum* azonosítója egy objektum-kijelölő objektumkifejezés. A megjegyzés a fejlesztési fázisban az EXE Options párbeszédpanelen lehet megadni.

CompanyName

Szintaxis: *objektum*.**CompanyName**

Ezt a futásidőben csak olvasható tulajdonságot **App** objektumon lehet használni a futó alkalmazás tulajdonosának vagy készítőjének nevét tartalmazó karakteres kifejezés kiolvasására. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A megjegyzés szövegét a programfejlesztési fázisban az EXE Options párbeszédpanelen lehet megadni.

ConflictTable

Szintaxis: *objektum*.**ConflictTable**

Ezt a tulajdonságot **TableDef** objektumon lehet használni és annak a melléktáblának a nevét adja vissza, amely két másolat szinkronizálása során az ütköző adatbázisrekordot tartalmazza. Ha nincs melléktábla vagy az adatbázis nem többszörözhető, a visszaadott érték üres ("") karakterlánc lesz. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Amennyiben az adatbázis ugyanazon rekordját két külön példány két felhasználója módosítja, az egyik felhasználó által végzett változtatás a másik példányon sikertelen lesz, ezért a sikertelen felhasználónak a problémát az ütközést feloldó eljárásnak kell átadnia. Az ütközések nem mezők között történnek, hanem rekordszinten következnek be.

A rekordok ütközése ütközési táblák létrehozásával kezelhető, amelyek a csere sikeres végrehajtása esetén a táblába helyezhető információkat tartalmaznak. Mindegyik melléktábla neve err_*xxxx* alakú, ahol az *xxxx* a tábla eredeti, a táblanév maximális hosszúságára szabott neve.

Connect

Szintaxis: *objektum*.**Connect** [= *adatbázistípus* ; [*paraméterek*;]

Ez a **Database**, **QueryDef** és **TableDef** objektumon, valamint **Data** vezérlőelemen érvényes tulajdonság egy megnyitott adatbázisra, átadott lekérdezésben használt adatbázisra vagy csatolt táblára vonatkozó információkat adó érték kiolvasására vagy beállítására használható. A tulajdonság a halmazhoz még nem csatolt **QueryDef**, **Database**, **TableDef** objektumok és csatolt táblák esetén írható és olvasható, alaptáblánál azonban csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A feltételes *adatbázistípus* argumentum egy, az adatbázis típusát megadó ka-

rakteres kifejezés. A Microsoft Jet adatbázis-szerkesztő esetén a ";" nélkül értendő, de a *paraméterek* argumentum használatakor, a ";" karaktert is meg kell adni.

- A feltételes *paraméterek* argumentum az ODBC vagy telepíthető ISAM vezérlők számára átadandó járulékos paramétereket tartalmazó karakteres kifejezés. Az egyes paramétereket ";" karakterekkel kell elválasztani egymástól.

A **Connect** tulajdonság az adatbázis típusjelzőjéből és egy vagy több, egymástól ";" karakterrel elválasztott paraméterből felépülő karakteres kifejezés, amelyet ODBC vagy telepíthető ISAM vezérlők számára járulékos paraméter átadására használhatunk. A Jet adatbázis-szerkesztőkhöz — a csatolt táblákat tartalmazók kivételével — nem használatos. Ilyenkor a **Connect** tulajdonság értéke üres ("") karakterlánc.

Egy .MDB fájlhoz csatolt táblán átadott lekérdezés végrehajtásához először a csatolt tábla adatbázisának **Connect** tulajdonságát egy érvényes ODBC kapcsolat-kifejezésre kell beállítani.

Csatolt táblát képviselő **TableDef** objektumnál a **Connect** tulajdonság egy vagy két részből (az adatbázis típusjelzőjéből és az adatbázis elérési útjából) állhat, az egyes részek után ";" karakterrel. Az elérési útnak az adatbázisfájlok teljes alakú elérési útjának kell lennie, amelyet a "DATABASE =" azonosítónak kell megelőznie. Néhány esetben — így a Jet, Btrieve és Excel adatbázisoknál — az adatbázis elérési út argumentum agy adott fájlnevet is magában foglal. A lehetséges adatbázistípusok és a nekik megfelelő típusjelzők és elérési utak a következők:

Adatbázistípus:	Típusjelző:	Elérési út:
Jet adatbázis	;	*meghajtó*:*elérési_út**fájlnév*.MDB
dBASE III	`dBASE III;`	*meghajtó*:*elérési_út*
dBASE IV	`dBASE IV;`	*meghajtó*:*elérési_út*
Paradox 3.x	`Paradox 3.x;`	*meghajtó*:*elérési_út*
Paradox 4.x	`Paradox 4.x;`	*meghajtó*:*elérési_út*
Btrieve	`Btrieve;`	*meghajtó*:*elérési_út**fájlnév*.DDF
FoxPro 2.0	`FoxPro 2.0;`	*meghajtó*:*elérési_út*
FoxPro 2.5	`FoxPro 2.5;`	*meghajtó*:*elérési_út*
FoxPro 2.6	`FoxPro 2.6;`	*meghajtó*:*elérési_út*
Excel 3.0	`Excel 3.0;`	*meghajtó*:*elérési_út**fájlnév*.XLS
Excel 4.0	`Excel 4.0;`	*meghajtó*:*elérési_út**fájlnév*.XLS
Excel 5.0	`Excel 5.0;`	*meghajtó*:*elérési_út**fájlnév*.XLS
Text	`Text;`	*meghajtó*:*elérési_út*
ODBC	`ODBC;`	Nincs

ODBC adatbázistípusnál a típusjelzőben megadható az alapértelmezett adatbázis, a felhasználó, jelszó, várakozási idő és más azonosító is. Ha csak táblázatban szereplő alakot használjuk, az összes regisztrált ODBC adatforrás nevét megadó párbeszédpanel az ODBC vezérlővel jelenik meg, így a felhasználó kiválaszthatja a kívánt adatbázist.

Ha szükséges jelszó, azonban a **Connect** tulajdonságban nincs megadva, az ODBC vezérlő táblához fordulásának első alkalmával, továbbá a kapcsolat bezárás utáni újra megnyitásakor a Login párbeszédpanel jelenik meg.

Egy **Database** objektum **Connect** tulajdonságát az **OpenDatabase** metódus feltételes *forrás* argumentumának megadásával lehet beállítani, amelyet az adatbázis típusának, elérési útjának, a felhasználó azonosítójának és jelszavának vagy ODBC adatforrásának meghatározásával lehet ellenőrizni.

A **QueryDef** objektumnál a **Connect** tulajdonságot a **ReturnsRecords** tulajdonsággal együtt egy ODBC SQL átadott lekérdezés létrehozására használható. A kapcsolatkifejezés *adatbázistípus* része `ODBC;` és a kifejezés további része a távoli adat eléréséhez az ODBC vezérlőre jellemző információ. Erről bővebb információt a vezérlő dokumentációja tartalmaz. A **Connect** tulajdonság beállításának meg kell előznie a **ReturnsRecords** beállítását.

Container

Szintaxis 1: *objektum*.**Container**

Szintaxis 2: *objektum*.**Container** [= *befogadó*]

Ezt a tulajdonságot a **Document** objektumon és a **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **Line**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox** és **VScrollBar** vezérlőelemeken eltérő szintaxissal és értelmezéssel lehet használni.

A **Document** objektumon a Szintaxis 1 érvényes, és ekkor a tulajdonság azt a **String** típusú **Container** objektumnevet adja vissza, amelyhez a **Document** objektum tartozik. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **Line**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox** vagy **VScrollBar** vezérlőelemeken a Szintaxis 2 érvényes, és ekkor a programfejlesztési fázisban nem elérhető tulajdonság egy **Form** objektumon lévő vezérlőelem befogadójának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *befogadó* argumentum más vezérlőelemek befogadójaként szolgáló objektum objektumkifejezése. Más objektum befogadója **Frame** és **PictureBox** vezérlőelem lehet.

ControlBox

Szintaxis: *objektum*.**ControlBox**

Ezt a futásidőben csak olvasható tulajdonság **Form** objektumon érvényes, és egy űrlapon a Control menü jelének megjelenését meghatározó érték kiolvasására használható. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **ControlBox** tulajdonság **Boolean** típusú visszaadott értékei és azok értelmezése a következő:

True Megjeleníti a Control menü jelét. (Alapértelmezett.)
False Eltávolítja a Control menü jelét.

A Control menü jelének megjelenítése a **BorderStyle** tulajdonság 1 (*Fixed Single*), 2 (*Sizable*) vagy 3 (*Fixed Dialog*) értékre állításával lehetséges.

A modális és nem modális ablakok egyaránt tartalmazhatják a Control menü jelét.

Futásidőben a kapcsolódó tulajdonságok értékétől függő menüparancsok lesznek elérhetők, például a **MaxButton** és **MinButton** tulajdonságok **False** értéke esetén a Control menuben nem jelennek meg a Maximize és Minimize parancsok, azonban a Move és a Close parancsok továbbra is elérhetők maradnak.

A **ControlBox**, **BorderStyle**, **MaxButton** és **MinButton** tulajdonságok beállításainak az űrlap megjelenésére csak futásidőben vannak hatással.

Copies

Szintaxis: *objektum*.**Copies** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Printer** objektumon és **CommonDialog** vezérlőelemen érvényes, és a nyomtatandó másolatok számának kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum a másolatok számát megadó **Integer** típusú numerikus kifejezés.

A Print párbeszédpanelen a tulajdonság a felhasználó által a Copies szövegbeviteli mező ben beállított másolatszámot adja vissza. Ha a **CommonDialog** vezérlőelemen a **UseDevModeCopies** jelző beállított, a **Copies** tulajdonság értéke mindig 1.

Count

Szintaxis: *objektum*.**Count**

A **Count** tulajdonságot adathozzáférési objektumokon és Visual Basic objektum-

gyűjteményeken azonos szintaxissal lehet használni. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság az összes adathozzáférési objektumon az objektumgyűjtemény objektumainak **Long** típusú számát adja vissza. A halmaz tagjainak számozása nullával kezdődik. A tulajdonság értéke sohasem lehet **Null**, ha a halmaz nem tartalmaz objektumot, a **Count** értéke 0 lesz.

Megjegyezzük, hogy a `For Each` *objektum* `In` *halmaz* szintaxis szerint is meg lehet határozni egy halmaz tagjainak számát.

A **Count** tulajdonság Visual Basic halmazokon, azaz a **Controls**, **ControlTemplates**, **Fields**, **Forms**, **Indexes**, **MenuItems**, **Properties**, **SelectedComponents**, **SelectedControls** és **TableDefs** objektumgyűjteményeken is érvényes, és itt is az objektumgyűjtemény objektumainak számát adja vissza.

A tulajdonságot a **For...Next** ciklussal együtt egy műveletnek az objektumgyűjtemény vezérlőelemein vagy űrlapjain történő végrehajtására, továbbá egy űrlap összes vezérlőelemeinek engedélyezésére vagy letiltására lehet használni.

A **Count** az **IfTypeOf** utasítással együtt lehetővé teszi, hogy a halmazban minden vezérlőelemen végiglépkedve azok egyes tulajdonságai módosítsuk.

A Visual Basic beépülő által megjelenített **SelectedComponents** halmaz **Count** tulajdonsága a Visual Basic 4.0-s verzióban nem lehet nagyobb 1-nél.

CurrentX

Szintaxis: *objektum*.**CurrentX** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Form** és **Printer** objektumon és **PictureBox** vezérlőelemen érvényes, és a következő nyomtatási vagy rajzolási metódus számára érvényes, az objektum bal oldali élétől mért vízszintes (X) koordináta kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum a vízszintes koordináta értékét megadó **Single** típusú szám.

A koordinátarendszer kezdőpontja az objektum bal felső sarka. Az *érték* alapértelmezetten twip, egyébként pedig a **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, **ScaleRight** és **ScaleMode** tulajdonságoknak megfelelő mértékegységben adott.

A grafikus metódusok a **CurrentX** tulajdonságot a következők szerint módosítják:

Ha a metódus:	A **CurrentX**:
Circle	Az objektum közepe.
Cls	Az $x = 0$ pont.
EndDoc	Az $x = 0$ pont.

Line	Az egyenes végpontja.
NewPage	Az $x = 0$ pont.
Print	A következő nyomtatási pozíció.
PSet	A rajzolt pont.

CurrentY

Szintaxis: *objektum*.**CurrentY** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Form** és **Printer** objektumon és **PictureBox** vezérlőelemen érvényes, és a következő nyomtatási vagy rajzolási metódus számára érvényes, az objektum felső élétől mért függőleges (Y) koordináta kiolvasására vagy beállítására alkalmas.

A további tudnivalók értelemszerűen azonosak a **CurrentX** tulajdonságnál leírtakkal.

Data

Szintaxis: *objektum*.**Data** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **OLE** befogadó vezérlőelemen érvényes, és egy adott formájú adatot tartalmazó memória vagy GDI (*Graphical Devive Interface*) objektum kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum a kezelőt meghatározó numerikus kifejezés. A kezelőhöz tartozó memóriát az *érték* = 0 beállítás szabadítja fel.

A tulajdonság használatával egy objektumot létrehozó alkalmazás számára lehet adatot küldeni, a memóriában vagy GDI objektumban lévő adat típusának a **Format** tulajdonságban történő beállítása után. Egy **OLE** objektumon érvényes formák listáját az **ObjectAcceptFormat** és **ObjectGetFormat** tulajdonságokkal lehet kiolvasni.

Az OLE Automatizmus könnyebb és megbízhatóbb módot nyújt egy objektumra vagy objektumról történő adat- és parancsküldésre. Az OLE Automatizmust támogató objektumhoz az **Object** tulajdonságán vagy a **CreateObject( )** és **GetObject( )** függvényeken keresztül lehet hozzáférni.

Database

Szintaxis: *objektum*.**Database**

Ez a **Data** vezérlőelemeken érvényes tulajdonság a vezérlőelem alapjául szolgáló **Database** objektum hivatkozását adja vissza. A tulajdonság szintaxisában az *objektum*

azonosítója egy objektumot kijelölő objektumkifejezés. A Visual Basic Standard változatában a **Database** tulajdonság a futásidőben csak olvasható, azonban a Professional Edition a programfejlesztési fázisban történő beállítására is lehetőséget ad.

A **Data** vezérlőelemmel létrehozott **Database** objektum a vezérlőelem **DatabaseName**, **Exclusive**, **ReadOnly** és **Connect** tulajdonságain alapul.

A **Database** objektum az adatokat tulajdonságain és metódusain keresztül szervezi, és bármely (például **Close** vagy **Execute**) metódusát a **Data** vezérlőelem **Database** tulajdonságával lehet használni. A **Database** belső szerkezete a **TablDefs** halmazzal, egyedi **TableDef** objektumé viszont a **Fields** és **Indexes** halmazokkal vizsgálható.

Bár létrehozhatunk **Recordset** objektumot és átadhatjuk számára egy **Data** vezérlőelem **Recordset** tulajdonságát, azonban az adatbázist nem lehet megnyitni és az újonan létrehozott **Database** objektumot nem lehet átadni a **Data** vezérlőelem **Database** tulajdonságának.

DatabaseName

Szintaxis: *objektum*.**DatabaseName** [= *elérési_út*]

A tulajdonság **Data** vezérlőelemeken érvényes a vezérlőelemhez tartozó adatforrás nevének és elérési útjának kiolvasására vagy beállítására. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *elérési_út* argumentum az adatbázisfájl helyét meghatározó karakteres kifejezés. Amennyiben a hálózat támogatja, a hálózati elérési utat például a következő alakban adhatjuk meg: "\\MYSERVER\MYSHARE\MYDIR\MYDB.MDB".

Az adatbázis típusát az *elérési_út* argumentumban adott fájl vagy könyvtár a következők szerint jelzi:

Ha az elérési_út:	Az adatbázis típusa:
.MDB fájl	Microsoft Access adatbázis.
.DDF fájl	Btrive adatbázis.
.DDF fájlt vagy fájlokat tartalmazó könyvtár	dBASE adatbázis.
.XLS fájlt tartalmazó könyvtár	Microsoft Excel adatbázis.
.DBF fájlt vagy fájlokat tartalmazó könyvtár	FoxPro adatbázis.
.PDX fájlt vagy fájlokat tartalmazó könyvtár	Paradox adatbázis.
Megfelelő adatbázisfájlokat tartalmazó könyvtár	Szövegformájú adatbázis.

Az ODBC adatbázisoknál (SQL Server vagy Oracle) a tulajdonság üresen hagyható, ha a vezérlőelem **Connect** tulajdonsága egyértelműen tartalmazza az ODBC.INI fájlban regisztrált adatforrás nevét (DSN).

Ha a vezérlőelem **Database** objektumának megnyitása után módosítjuk a **DatabaseName** tulajdonságot, az új adatbázist a **Refresh** metódussal kell megnyitni.

A külső adatbázisok használatakor a jobb teljesítmény eléréséhez célszerű a külső adatbázisok tábláit egy Microsoft Jet adatbázishoz (.MDB) csatolni és a **DatabaseName** tulajdonságban ezen adatbázisnevet használni.

DataChanged

Szintaxis: *objektum*.**DataChanged** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **CheckBox**, **DBCombo**, **DBList**, **Image**, **Label**, **PictureBox** és **TextBox** vezérlőelemeken érvényes, és használatával az azt jelző értéket lehet kiolvasni vagy beállítani, hogy a kötött vezérlőelemben az adatot az aktuális rekordból az adatot kiolvasótól különböző eljárás módosította-e. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum az adat változására jellemző **Boolean** típusú érték.

Az *érték* argumentum értéke és értelmezése a következő lehet:

True A vezérlőelemben lévő aktuális adat nem azonos az aktuális rekordban lévővel.

False A vezérlőelemben lévő aktuális adat (ha van), azonos az aktuális rekordban lévővel. (Alapértelmezett.)

Amikor egy **Data** vezérlőelem az egyik rekordról a másikra lép, adatot ad át az aktuális rekord mezőiből az adott mezőhöz vagy a teljes rekordhoz kötött vezérlőelemeknek. A kötött vezérlőelemekben történő adatmegjelenítés **False** értékre állítja a **DataChanged** tulajdonságot. Ha ezt követően a kötött vezérlőelemben az értéket a felhasználó vagy bármely más művelet módosítja, a **DataChanged** tulajdonság értéke **True** lesz. Másik rekordra történő átlépés egymagában nem hat a **DataChanged** értékére.

A **Data** vezérlőelem másik rekordra lépésekor egy **Validate** esemény következik be. Ha bármely kötött vezérlőelem **DataChange** tulajdonsága **True**, a vezérlőelem automatikusan meghívja az **Edit** és **UpdateRecord** metódusokat a változások adatbázisba továbbítására. Ezt a működést ismerve a **Validate** eseményeljárásban egy adatbázishoz kötött vezérlőelem **DataChanged** tulajdonságát **False** értékre állítva letiltható a vezérlőelemen lévő adatnak az adatbázisba való mentése.

A kaszkád események elkerülése érdekében javasolt a kötött és kötetlen vezérlőelemek **Change** eseményeljárásában ellenőrizni a **DataChange** tulajdonság értékét.

DataField

Szintaxis: *objektum*.**DataField** [= *érték*]

Ez a tulajdonság **CheckBox**, **ComboBox**, **DBCombo**, **DBGrid**, **DBList**, **Image**,

Label, **ListBox**, **PictureBox** és **TextBox** vezérlőelemeken érvényes, és egy vezérlőelemet az aktuális rekord egy mezőjéhez kötő értéket lehet kiolvasni vagy beállítani. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum egy karakteres kifejezés, amely a **Data** vezérlőelem **RecordSource** és **DatabaseName** tulajdonságaival meghatározott **Rekordset** objektumban a mezők egyikének nevét határozza meg.

Az adatbázishoz kötött vezérlőelemek az adatbázisokban meghatározott adatokhoz való hozzáférést tesznek lehetővé. Az egyetlen mezőt kezelő kötött vezérlőelemek jellemzően az aktuális rekord egy adott mezőjének értékét jelenítik meg.

Egy kötött vezérlőelem **DataSource** tulajdonsága egy érvényes **Data** vezérlőelem nevét, a **DataField** pedig egy, a **Data** vezérlőelem által létrehozott **Recordset** objektumban érvényes mezőnevet határoz meg, így együttesen kijelölve a kötött vezérlőelemben megjelenő adatot.

Egy kifejezés értékét visszaadó **QueryDef** objektum vagy SQL utasítás használatakor a Microsoft Jet automatikusan állítja elő a mezőnevet. Általában a kifejezés mezőneve Expr1*xxx* alakú, ahol az *xxx* egy 000-tól kezdődő sorszám, azaz az első visszaadott kifejezés neve Expr1000 lesz.

Mindegyik kötött vezérlőelemnél ellenőrizni kell, hogy a **DataField** tulajdonsága érvényes beállítás-e. Ha a **Data** vezérlőelem **RecordSource** tulajdonságának módosítását követően használjuk a **Refresh** metódust, a **Recordset** az új objektumot fogja használni, ezzel érvénytelenné téve a kötött vezérlőelem **DataField** értékeit és kezelhető hibát okozva.

DataSource

Szintaxis:

Ez a futásidőben nem elérhető tulajdonság **CheckBox**, **ComboBox**, **DBCombo**, **DBGrid**, **DBList**, **OLE** befogadó, **Image**, **Label**, **ListBox**, **PictureBox** és **TextBox** vezérlőelemeken érvényes, és a **Data** vezérlőelemet az egyik aktuális vezérlőelemének adatbázishoz kötésén keresztül határozza meg.

Egy vezérlőelemnek az adatbázis egy mezőjéhez futásidőben történő kötéséhez a **Data** vezérlőelemet a **DataSource** tulajdonságban már a programfejlesztési fázisban meg kell adni a Properties ablakban. A **Data** vezérlőelemmel szervezett **Recordsert** objektum mezőjével való összekapcsolás a **DataField** tulajdonságban a **Field** objektum megnevezésével tehető befejezetté. A **DataField** tulajdonság beállítása — eltérően a **DataSource** tulajdonságtól — futásidőben is megengedett.

DataText

Szintaxis: *objektum*.**DataText** [= *érték*]

Ez a tulajdonság **OLE** befogadó vezérlőelemeken érvényes, és egy adott objektumhoz egy karakterlánc beállítására vagy onnan egy karakterlánc kiolvasására hasznnálható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum egy karakteres kifejezés, amely az objektumhoz tartozó karakterláncot tartalmazza.

Egy karakterláncnak egy objektum számára történő küldéséhez előzőleg a **Format** tulajdonságot az objektum által támogatott formának megfelelőre kell beállítani. Az objektum által támogatott formátumok listája az **ObjectGetFormat** és **ObjectAcceptFormat** tulajdonságokkal olvasható ki.

Ha az objektumból adatot kívánunk olvasni, a **DataText** tulajdonság egy, az objektum által küldött, az első 0 karakterrel lezárt karakterláncot ad vissza. A Visual Basic 16-bites verziójában a karakterlánc nem haladhatja meg a 64K méretet, a 32-bites verzióban pedig a rendelkezésre álló memória által megengedett méretet.

Az OLE Automatizmus könnyebb és megbízhatóbb módot nyújt egy objektumra vagy objektumról történő adat- és parancsküldésre. Az OLE Automatizmust támogató objektumhoz az **Object** tulajdonságán vagy a **CreateObject( )** és **GetObject( )** függvényeken keresztül lehet hozzáférni.

DataUpdatable

Szintaxis: *objektum*.**DataUpdatable**

Ez a **Field** objektumon érvényes, mindig csak olvasható tulajdonság az objektum által képviselt mezőbeni adat frissíthetőségét jelző értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **DataUpdatable** által visszaadott érték és annak értelmezése a következő lehet:

True A mezőbeni adat frissíthető.
False A mezőbeni adat nem frissíthető.

A tulajdonságot annak meghatározására lehet használni, hogy a **Field** objektum **Value** tulajdonságának értéke módosítható-e. A **DataUpdatable** nem támogatott **Index** és **Relation** objektumok **Fields** halmazához csatolt **Field** objektumokon.

DateCreated

Szintaxis: *objektum*.**DateCreated**

A tulajdonság **Document**, **QueryDef**, tábla típusú **Recordset**, **TableDef** (valamint az elavult **Table**) objektumon érvényes, és az objektumon alapuló alaptábla létrehozásának dátumát és időpontját a **Variant** adattípus **VarType** 7 (Date) altípusa szerinti értékként adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A dátum és az időpont értékek abból a számítógépből származnak, amelyen az alaptábla létrehozása történt. A többfelhasználós környezetben a felhasználók a **DateCreated** tulajdonság értékét közvetlenül a fájlkiszolgálótól kaphatják meg.

Default

Szintaxis: *objektum*.**Default** [= *érték*]

A tulajdonság **CommandButton** és **OLE** befogadó vezérlőelemeken érvényes, és egy űrlapon a parancsgomb alapértelmezett gombkénti használatára jellemző érték beállítására vagy kiolvasására alkalmas. A parancsgomb **CommandButton** vezérlőelem vagy egy **OLE** befogadó vezérlőelemen belül bármely, parancsgombként funkcionáló objektum lehet. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

A tulajdonság **Boolean** típusú értéke és annak értelmezése a következő lehet:

True	A **CommandButton** vezérlőelem az alapértelmezett gomb.
False	A **CommandButton** vezérlőelem nem az alapértelmezett gomb. (Alapértelmezett.)

Egy űrlapon csak egyetlen **CommandButton** vezérlőelem lehet alapértelmezett parancsgomb. Amikor egy gomb **Default** tulajdonsága **True**, az összes többi parancsgomb ugyanezen tulajdonsága automatikusan **False** értékűvé válik. Ha egy aktív űrlap **CommandButton** vezérlőelemének **Default** tulajdonsága **True**, a felhasználó ezt a parancsgombot az ENTER billentyű leütésével — és ezzel egy **Click** eseményt okozva — választhatja ki. Más fókuszban lévő vezérlőelem nem kaphatja meg a billentyűzet eseményeit (**KeyDown**, **KeyPress** és **KeyUp**), kivéve, ha a fókuszt ugyanazon űrlap másik gombjára visszük át, ekkor az ENTER billentyű leütése az alapértelmezett gomb helyett a fókuszban álló gombot teszi kiválasztottá.

Az **OLE** befogadó vezérlőelem esetén a **Cancel** tulajdonság csak azon objektumok-

nál használható, amelyek parancsgombként funkcionálnak.

A **Default** tulajdonságot a **Cancel** tulajdonsághoz hasonlóan elsősorban akkor használjuk, ha egyes parancsgombok véletlen használatát ki kívánjuk zárni. Egy végzetes, visszaállíthatatlan hatású, például fájlokat törlő műveleti parancsot támogató űrlap vagy párbeszédpanel Cancel gombját a **Default** tulajdonságának **True** értékre beállításával lehet alapértelmezetté tenni.

DefaultExt

Szintaxis: *objektum*.**DefaultExt**[= *érték*]

A tulajdonság **CommonDialog** vezérlőelemen érvényes, és a párbeszédablakon alapértelmezetten érvényes fájlnév-kiterjesztés beállítására vagy kiolvasására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a fájlnév-kiterjesztést tartalmazó karakteres kifejezés.

A tulajdonságot az alapértelmezett kiterjesztés, például .EXE vagy .DOC meghatározására lehet használni. Ha a fájl kiterjesztés megadása nélkül van mentve, a fájlnévhez automatikusan hozzáadódik a **DefaultExt** tulajdonságban szereplő kiterjesztés.

DefaultUser

Szintaxis: *objektum*.**DefaultUser** [= *érték*]

A tulajdonság **DBEngine** objektumon érvényes, és a Microsoft Jet adatbázis-szerkesztő által használt egy **Workspace** objektum azonosítónév nélküli létrehozásakor az alapértelmezett felhasználói azonosítónév beállítására. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy érvényes felhasználói nevet tartalmazó, **String** típusú karakteres kifejezés. A név 1-20 karakter hosszúságú lehet, és tartalmazhat normál és ékezetes betűket, számokat, szóközt és a következők kivételével szimbólumokat is: idézőjel ("), törtvonal (/), fordított törtvonal, backslash (\), szögletes idézőjelek ([és]), kettőspont (:), pipa (|), kisebb mint jel (<), nagyobb mint jel (>), plusz jel (+), egyenlőségjel (=), pontosvessző (;), vessző (,), kérdőjel (?), csillag (*), bevezető szóköz, valamint a vezérlő karakterek (ASCII 00 — ASCII 31).

A felhasználói nevekben a kis- és nagybetűk általában egyenértékűek, azonban egy előzőleg törölt vagy másik munkacsoportban létrehozott név létrehozásakor a felhasz-

nálói névnek tökéletesen egyezőnek kell lennie az eredeti névvel.

A **CreateWorkspace** metódus jellemzően **Workspace** objektumok felhasználói névvel és jelszóval történő létrehozására használatos, azonban az előző verziókkal való kompatibilitás miatt a Microsoft Jet azt szükség esetén — ha még nem lenne egy **Workspace** nyitott — automatikusan is létrehozza a **DefaultUser** és **DefaultPassword** értékek alapján.

Az alapértelmezett **UserName** és **Password** tulajdonságok nem lesznek érintettek, ha a **Workspace** objektumra a **DefaultUser** és **DefaultPassword** tulajdonságok beállítása előtt hivatkozunk.

DefaultValue

Szintaxis: *objektum*.**DefaultUser** [= *érték*]

A tulajdonság **Field** objektumon és **DBGrid** vezérlőelemen érvényes, és egy objektum alapértelmezett értékének beállítására vagy kiolvasására alkalmas. **Fields** objektumgyűjteményhez még nem tartozó objektum esetén a tulajdonság írható és olvasható is, egyébként használata az objektumgyűjteményt tartalmazó objektumtól függ. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **String** típusú, legfeljebb 255 karaktert tartalmazó szöveg vagy felhasználó által definiált függvényt, Microsoft Jet összesítő függvényeket, SQL összesítő függvényeket vagy lekérdezésre, űrlapokra vagy más **Field** objektumokra vonatkozó hivatkozásokat nem tartalmazó karakteres kifejezés.

Field objektumnál a **DefaultValue** tulajdonság használata attól a **Fields** halmazt tartalmazó objektumtól függ, amelyhez a **Field** tartozik:

Ha az objektum:	A tulajdonság használata:
Index	Nem támogatott.
QueryDef	Csak olvasható.
Recordset	Csak olvasható.
Relation	Nem támogatott.
TableDef	Írható és olvasható.

Új rekord létrehozásakor a **DefaultValue** beállítása automatikusan a mező értékévé válik, amelyet a **Value** tulajdonsággal lehet módosítani.

A **DefaultValue** tulajdonság nem használható a **Counter** és **OLE** objektum mezőkön.

DefaultColWidth

Szintaxis: *objektum*.**DefaultColWidth** [= *érték*]

A tulajdonság **DBGrid** vezérlőelemen érvényes, és az objektum mindegyik oszlopára érvényes alapértelmezett oszlopszélesség értékének beállítására vagy kiolvasására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Integer** típusú szám, amely a **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, **ScaleRight** és **ScaleMode** tulajdonságoknak megfelelő mértékegységben értelmezett. Ha az *érték* = 0, az összes oszlop szélessége automatikusan az oszlop fejlécének szélessége vagy az alapjául szolgáló mező **Size** tulajdonságában adott érték közül a nagyobbik lesz.

DefaultPassword

Szintaxis: *objektum*.**DefaultPassword** [= *érték*]

A tulajdonság **DBEngine** objektumon érvényes, és a Microsoft Jet adatbázisszerkesztő által használt egy **Workspace** objektum meghatározott jelszó nélküli létrehozásakor az alapértelmezett felhasználói jelszó beállítására. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy érvényes jelszót tartalmazó, **String** típusú karakteres kifejezés. A jelszó legfeljebb 14 karakter hosszúságú lehet, és az ASCII 0 kivételével bármely karaktert tartalmazhat.

A **CreateWorkspace** metódus jellemzően **Workspace** objektumok felhasználói névvel és jelszóval történő létrehozására használatos, azonban az előző verziókkal való kompatibilitás miatt a Microsoft Jet azt szükség esetén — ha még nem lenne egy **Workspace** nyitott — automatikusan is létrehozza a **DefaultUser** és **DefaultPassword** értékek alapján.

Az alapértelmezett **UserName** és **Password** tulajdonságok nem lesznek érintettek, ha a **Workspace** objektumra a **DefaultUser** és **DefaultPassword** tulajdonságok beállítása előtt hivatkozunk.

Description

Szintaxis: *objektum*.**Description** [= *érték*]

A tulajdonság **ClassModule**, **Error** és **Err** objektumon eltérő módon érvényes.

A **ClassModule** objektumon a tulajdonság egy osztály kiválasztásakor az Object

Browser által megjelenített, az osztály használatát, tulajdonságait és metódusait ismertető szöveget tartalmazza. A tulajdonság alapértelmezetten üres ("") karakterlánc, amelyet az Object Browser listájában a projekt és osztály kiválasztása után az Options gombra kattintva megjelenítetett Member Options párbeszédpanel Description szövegbeviteli mezőjében lehet módosítani. A **Description** tulajdonság beállítása után az adott osztály tulajdonságait és metódusait az Object Browser használatával a többi felhasználó is megismerheti.

Meg kell jegyezni, hogy a **Description** tulajdonság nem azonos a Project Options párbeszédpanelen megadott Application Description karakterlánccal, amely a References párbeszédpanelben szereplő **OLE** kiszolgálót határozza meg.

Az **Error** és **Err** objektumon a tulajdonság egy hibához kapcsolódó ismertető szöveg kiolvasására és beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a hiba leírását tartalmazó karakteres kifejezés.

A **Description** tulajdonságban egy hiba rövid leírását tartalmazza és megjelenítésével a felhasználót figyelmeztetni lehet egy bekövetkezett, de valamely okból nem lekezelt hibára.

A **Description** tulajdonságnak az **Err** objektummal történő használatkor a felhasználó által definiált hiba generálásakor egy rövid hibaleírást lehet a tulajdonságba beállítani. Ha a **Description** tulajdonság értéke üres és a **Number** tulajdonság egy Visual Basic futásidejű hibaszámnak felel meg, egy hiba bekövetkeztekor a **Description** tulajdonságba az **Error( )** függvény által visszaadott karakterlánc kerül.

DesignMasterID

Szintaxis: *objektum*.**DesignMasterID** [= *érték*]

A tulajdonság **Database** objektumon érvényes, és az alappéldányt szolgáltató, a Microsoft Jet által előállított 16-bites egyedi azonosító (OLE 2 GUID) értékének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az adatbázis-objektumot alappéldányként kijelölő 16-bites **ReplicaID**.

A **DesignMasterID** értékének tárolása az MSysReplica rendszertáblában történik. A tulajdonságnak egy másik adatbázis **ReplicaID** értékét adva a másolathalmazban az az adatbázis kapja meg az alappéldány rangját. A **DesignMasterID** beállítása csak az aktuális alappéldányon lehetséges, egyébként két összeegyeztethetetlen, szinkronizálha-

tatlan halmaz jön létre. Ha az eredeti példány elveszett vagy törlődött, egy másolaton történő **DesignMasterID** beállítás az aktuális másolatot teszi az új alappéldánnyá.

DeviceName

Szintaxis: *objektum*.**DeviceName**

A tulajdonság **Printer** objektumon érvényes, és egy vezérlőprogram által támogatott eszköznevet — például HP LaserJet III Si — ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Minden nyomtatóvezérlő egy vagy több eszközt támogathat. A **Printer** objektum tulajdonságainak hatása a nyomtató gyártója által adott vezérlőprogramtól függ: egyes tulajdonságok hatástalanok lehetnek vagy több tulajdonság azonos hatást eredményezhet. Erre vonatkozóan a nyomtató leírása tartalmazhat további információt.

DialogTitle

Szintaxis: *objektum*.**DialogTitle** [= *érték*]

A tulajdonság **CommonDialog** vezérlőelemen érvényes, és egy párbeszédpanel címsorában megjelenő szöveg kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a párbeszédpanel címsorában megjelenő szöveget tartalmazó karakteres kifejezés. Az alapértelmezett címfelirat az Open párbeszédpanelen "Open", a Save As párbeszédpanelen "Save As".

A **CommonDialog** vezérlőelem a Color vagy Font párbeszédablak megjelenítése esetén figyelmen kívül hagyja a **DialogTitle** tulajdonság beállítását.

DisplayType

Szintaxis: *objektum*.**DisplayType** [= *érték*]

A tulajdonság **OLE** befogadó vezérlőelemen érvényes, és egy objektumnak a tartalmát vagy az ikonját megjelenítő állapotára jellemző érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy szimbolikus konstans vagy **Integer** típusú érték, amely az objektum tartalmának vagy ikon-

jának megjelenítését határozza meg.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	A megjelenítés:
vbOLEDisplayContent	0	*Content* — Az objektumot tartalmazó OLE vezérlőelemben az objektum adata jelenik meg. (Alapértelmezett.)
vbOLEDisplayIcon	1	*Icon* — Az objektumot tartalmazó OLE vezérlőelemben az objektum ikonja jelenik meg.

A **DisplayType** tulajdonság értéke az Insert Object és a Paste Special párbeszédpaneleken megjelenő Display As Icon választómező alapértelmezett beállítását is meghatározza. Ha a **DisplayType** értéke 1 (*Icon*), az Insert Object és a Paste Special párbeszédpanelek programfejlesztési időben vagy az **InsertObjDlg** vagy **PasteSpecialDlg** metódusokkal futásidőben történő megjelenítésekor a Display As Icon választómező automatikusan kiválasztott lesz.

A **DisplayType** tulajdonsággal a futásidőben a **CreateEmbed** vagy **CreateLink** metódussal létrehozott objektumok megjelenítési típusa is beállítható, azonban a már létrehozott objektumok megjelenítési típusa nem változtatható meg.

DistinctCount

Szintaxis: *objektum*.**DistinctCount**

A tulajdonság **Index** objektumon érvényes, és az objektum egyedi — a hozzá kapcsolt táblában található — értékeinek számát jelző, **Long** típusú értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonságot egy indexben lévő egyedi értékek vagy kulcsok számának meghatározására használhatjuk. Egy kulcs még akkor is csak egyszer kerül figyelembevételre, ha az érték többszörös előfordulása is megengedett. A **DistinctCount** egy változást követően nem mindig azonnal tükrözi az aktuális kulcsszámot, például egy tranzakció visszavonásának hatását.

A tulajdonság által adott információ olyan alkalmazásokban hasznos, amelyek az indexinformációk alapján optimálják az adathozzáférést.

DividerStyle

Szintaxis: *objektum*.**DividerStyle** [= *érték*]

A tulajdonság **Column** objektumon érvényes, és egy **DBGrid** vezérlőelemben kiválasztott oszlop jobb oldali keretvonalának stílusát jellemző érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argu-

mentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a keret stílusát meghatározó egész szám.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
0	Nincs elválasztóvonal.
1	Fekete vonal. (Alapértelmezett.)
2	Sötétszürke vonal.
3	Kiemelt hatású.
4	Besüllyesztett hatású.
5	Az elválasztóvonal a **ForeColor** tulajdonság szerinti színnel rajzolt.

A **DividerStyle** tulajdonság nem befolyásolja, hogy a keret elhúzható-e vagy sem. A kiemelt vagy besüllyesztett keret esetén használt színt a Windows határozza meg.

DragIcon

Szintaxis: *objektum*.**DividerStyle** [= *ikon*]

A tulajdonság **CheckBox**, **ComboBox**, **CommandButton**, **Data**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemen érvényes, és az "elhúzás-és-leejtés" módszer használatakor az egér mutatójaként megjelenő ikon kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *ikon* argumentum bármely, ikont visszaadó kódhivatkozás lehet, például egy űrlap ikonjára történő hivatkozás (`Form1.Icon`), másik vezérlőelem DragIcon tulajdonságára történő hivatkozás (`Text1.DragIcon`) vagy **LoadPicture( )** függvény.

Az *ikon* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *ikon*:	A mutató:
(nincs)	Egy négyzeten belüli nyíl. (Alapértelmezett.)
Icon	Felhasználó által definiált, a programfejlesztési fázisban a Properties ablakban, futásidőben a **LoadPicture( )** függvénnyel megadható egérmutató. A betöltendő fájlnak .ICO kiterjesztésűnek és formájúnak kell lennie.

A **DragIcon** tulajdonság vizuális visszacsatolási célra használható egy elhúzási mű-

velet során, például így jelezve, hogy az egérmutató a megfelelő célobjektum felett van. A tulajdonság akkor hatásos, ha a felhasználó egy "elhúzás-és-leejtés" műveletet indít el, vagyis jellemzően egy **MouseDown** vagy **DragOver** eseményeljárás része lehet.

Futásidőben a tulajdonság bármelyik objektum **DragIcon** vagy **Icon** tulajdonságához vagy a **LoadPicture( )** függvény által visszaadott ikonhoz is beállítható.

DragMode

Szintaxis: *objektum*.**DragMode** [= *érték*]

A tulajdonság **CheckBox**, **ComboBox**, **CommandButton**, **Data**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemen érvényes, és az "elhúzás-és-leejtés" módszer használatakor a vezérlőelem automatikus vagy kézi megfogási módját meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a megfogási módot meghatározó egész szám.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	A megfogás módja:
0	*Manual* — A forrás vezérlőelem megfogásához a **Drag** metódus használata szükséges. (Alapértelmezett.)
1	*Automatic* — A forrás vezérlőelemen az egérkattintás automatikusan egy **Drag** metódust aktivál. **Az OLE** befogadó vezérlőelem csak akkor válik automatikusan megfogottá, ha a fókuszban van.

Amikor a tulajdonság beállítása 1 (*Automatic*), a vezérlőelem nem válaszol a szokásos módon az egéreseményekre, ezért a 0 (*Manual*) beállítással kell meghatározni az "elhúzás-és-leejtés" művelet kezdetét és befejezését: a beállítással egy billentyűzet vagy menüparancs indíthatja a műveletet vagy engedélyezi egy forrás vezérlőelemnek a művelet előtt a **MouseDown** esemény felismerését.

Az "elhúzás-és-leejtés" művelet alatt a célobjektumra kattintás a célobjektumon egy **DragOver**, majd egy **DragDrop** eseményt okoz és befejezi a műveletet.

Egy vezérlőelem az elhúzása közben nem képes felismerni a felhasználó által végrehajtott egér- vagy billentyűzeteseményeket (**KeyDown**, **KeyPress**, **KeyUp**, **MouseDown**, **MouseMove** és **MouseUp**), de a programkód vagy DDE kapcsolat által okozott eseményeket igen.

DrawMode

Szintaxis: *objektum*.**DrawMode** [= *érték*]

A tulajdonság **Form** és **Printer** objektumon, **Line**, **PictureBox** vagy **Shape** vezérlőelemen érvényes, és a grafikus metódusok eredményének vagy a **Line** és **Shape** vezérlőelemek megjelenését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a megjelenítési módot meghatározó egész szám.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	A megjelenési mód:
1	*Blackness* — Fekete.
2	*Not Merge Pen* — A **DrawMode** = 15 beállítás (*Merge Pen*) inverze.
3	*Mask Not Pen* — A hátér- és az inverz rajzszín közös színeinek kombinációja.
4	*Not Copy Pen* — A **DrawMode** = 13 beállítás (*Copy Pen*) inverze.
5	*Mask Pen Not* — A rajz- és az inverz képernyőszín közös színeinek kombinációja.
6	*Invert* — Az inverz képernyőszín.
7	*Xor Pen* — A rajz- és a képernyőszín nem közös színeinek kombinációja.
8	*Not Mask Pen* — A **DrawMode** = 9 beállítás (*Mask Pen*) inverze.
9	*Mask Pen*- A rajz- és a képernyőszín közös színeinek kombinációja.
10	*Not Xor Pen* — A **DrawMode** = 7 beállítás (*Xor Pen*) inverze.
11	*No operation* — Nincs művelet, a megjelenítés változatlan, a rajzolást kikapcsolja.
12	*Merge Not Pen* — A megjelenítés a képernyő- és az inverz rajzszín kombinációja.
13	*Copy Pen* — A **ForeColor** tulajdonságban adott rajzszín. (Alapértelmezett.)
14	*Merge Pen Not* — A rajz- és az inverz képernyőszín kombinációja.
15	*Merge Pen* — A rajz- és a képernyőszín kombinációja.
16	*Whiteness Pen* — Fehér.

A tulajdonsággal a **Line** vagy **Shape** vezérlőelemek vagy a grafikus metódusok eredményeinek vizuális hatásait lehet beállítani. A Visual Basic a rajz és a hozzá tartozó háttér minden egyes képpontját összehasonlítja, majd az *érték* beállítástól függően bitenként kombinálja ezek színét. A **DrawMode** beállításának hatása a futásidőben rajzolt vonal és a háttér színének kombinációs módjától függ. Az 1, 6, 7, 11, 13 és 16 *érték* adja a legjobban előrebecsülhető eredményt.

DrawStyle

Szintaxis: *objektum*.**DrawStyle** [= *érték*]

A tulajdonság **Form** és **Printer** objektumon, valamint **PictureBox** vezérlőelemen érvényes, és a grafikus metódusok rajzolási stílusát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a rajzolási stílust meghatározó egész szám.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	A rajzolási stílus:
0	*Solid* — Folytonos vonal. (Alapértelmezett.)
1	*Dash* — Szaggatott vonal. (— - — - —)
2	*Dot* — Pontozott vonal. (.....)
3	*Dash-dot* — Pontvonalas vonal. (.-.-.-.-)
4	*Dash-dot-do* — Kettős-pontvonalas vonal. (..-..-..)
5	*Transparent* — Átlátszó vonal.
6	*Inside solid* — Kitöltött folytonos vonal.

A **DrawStyle** tulajdonság csak a **DrawWidth** = 1 beállítás mellett adja a felsorolt rajzolási stílust, egyébként a **DrawStyle** = 1...4 beállítás hatástalan, a vonal folytonos lesz.

DrawWidth

Szintaxis: *objektum*.**DrawWidth** [= *érték*]

A tulajdonság **Form** és **Printer** objektumon, valamint **OLE** befogadó és **PictureBox** vezérlőelemen érvényes, és a grafikus metódusok rajzolási vonalvastagságát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a rajzolási vonalvastagságot meghatározó, 1 (alapértelmezett) és 32767 közötti egész számot adó numerikus kifejezés. A vonalvastagság pixel egységben értendő.

A **DrawWidth** nem alapértelmezett (1) értékei mellett a rajzolt vonal megjelenése a **DrawStyle** 1...4 beállítástól függetlenül folytonos lesz.

Drive

Szintaxis: *objektum*.**Drive** [= *meghajtó*]

Ez a programfejlesztési fázisban nem használható tulajdonság **DriveListBox** vezérlőelemen érvényes, és a futásidőben kiválasztott meghajtó kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *meghajtó* argumentum a kiválasztott — a vezérlőelem létrehozásakor vagy futásidejű frissítésekor rendelkezésre álló vagy a rendszerhez csatlakoztatott — bármelyiket meghatározó karakteres kifejezés. Az alapértelmezett mindig az éppen érvényes meghajtó.

A **Drive** tulajdonság által visszaadott érték a következő formák valamelyike lehet:

- Floppy-meghajtóknál: `"a:"` vagy `"b:"`
- Keménylemezes meghajtóknál: `"c: [ kötetazonosító ]"`
- Hálózati kapcsolatnál: `"x:\\kiszolgalo\megosztas"`

A tulajdonság beállításnál a következőkre kell figyelemmel lenni:

- A meghajtónév a *meghajtó* karakteres kifejezés első karaktere lesz. A meghajtónévben a kis- és nagybetűs alak egyenértékű.
- A **Drive** tulajdonság beállításának módosítása egy **Change** eseményt okoz.
- Nem létező vagy nem elérhető meghajtó kijelölése hibát okoz.
- A tulajdonság beállítása a meghajtólistát is frissíti, így a programkódon belül módot adva a vezérlőelem létrehozása óta hozzáadott hálózati kapcsolatok követésére.

Ha a **FileName** tulajdonság beállításában megadott elérési út nem tartalmaz meghajtónevet, a **Drive** tulajdonság értékébe üres karakterlánc ("") és a **ListIndex** tulajdonság értékébe -1 kerül, valamint nem lesz kiválasztott meghajtó.

DriverName

Szintaxis: *objektum*.**DriverName**

Ez a **Printer** objektumon és **Printers** objektumgyűjteményen érvényes tulajdonság egy objektum vezérlőprogramjának nevét adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Minden nyomtató-vezérlőprogramnak egyedi neve van, például számos Hewlett-Packard nyomtató esetén a **DriverName** tulajdonság HPPCL5MS lesz. A tulajdonság jellemzően egy kiterjesztés nélküli fájlnév.

A **Printer** objektum tulajdonságainak hatása a nyomtató gyártója által adott vezérlőprogramtól függ: egyes tulajdonságok hatástalanok lehetnek vagy több tulajdonság azonos hatást eredményezhet. Erre vonatkozóan a nyomtató leírása tartalmazhat további információt.

Duplex

Szintaxis: *objektum*.**Duplex** [= *érték*]

Ez a **Printer** objektumon és **Printers** objektumgyűjteményen érvényes, programfejlesztési fázisban nem elérhető tulajdonság a kétoldalas nyomtatást támogató nyomtatókon a nyomtatási módot meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *meghajtó* argumentum a nyomtatási módot meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	A nyomtatási mód:
vbPRDPSimplex	1	Egyoldalas, az aktuális orientációnak megfelelő nyomtatás.
vbPRDPHorizontal	2	Kétoldalas, vízszintes lapfordítást használó nyomtatás.
vbPRDVertical	3	Kétoldalas, függőleges lapfordítást használó nyomtatás.

A kétoldalas, vízszintes lapfordítást használó nyomtatásnál az oldalak teteje a lap ugyanazon végére kerül, a függőleges lapfordításnál pedig a lap egyik végére az egyik oldal alja és a következő teteje kerül.

A **Printer** objektum tulajdonságainak hatása a nyomtató gyártója által adott vezérlőprogramtól függ: egyes tulajdonságok hatástalanok lehetnek vagy több tulajdonság azonos hatást eredményezhet. Erre vonatkozóan a nyomtató leírása tartalmazhat további információt.

EditMode

Szintaxis: *objektum*.**EditMode**

Ez a **Data** vezérlőelemen, valamint eredményhalmaz és tábla típusú **Recordset** objektumon érvényes tulajdonság az aktuális rekord szövegbeviteli állapotát jelző értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság által visszaadott értékek és értelmezésük a következő lehet:

Ha az EditMode:	A szövegbeviteli állapot:
vbEditNone	Nincs folyamatban szövegbeviteli művelet.
vbEditInProgress	Az **Edit** metódus meghívása megtörtént és az aktuális rekord a másolatpufferbe került.
vbEditAdd	Egy **AddNew** metódus meghívása történt, és a másolatpufferben lévő aktuális rekord olyan új rekord, amely még nem volt az adatbázisba mentve.

Az **EditMode** tulajdonság a **Data** vezérlőelem alapértelmezett működésmódjától történő eltérés vagy a **Data** vezérlőelemnek nem Visual Basic Professional Edition változatban használható előnyösen: az **Update** metódus meghívhatóságának eldöntéséhez a **Validate** eseményeljárásban a *művelet* argumentum és az **EditMode** értéke, az aktuális adatlap zároltsági állapotának ellenőrzéséhez a **True** értékű **LockEdits** és a **dbEditInProgress** értékű **EditMode** ad lehetőséget.

Enabled

Szintaxis: *objektum*.**Enabled** [= *érték*]

Ez a **Form** és **MDIForm** objektumon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **Data**, **DBCombo**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **Menu**, **MenuLine**, **OLE** befogadó, **OptionButton**, **PictureBox**, **TextBox**, **Timer** és **VScrollBar** vezérlőelemen érvényes tulajdonság az űrlap vagy vezérlőelem felhasználó által előidézett eseményre — például egy billentyű vagy egérgomb lenyomására — való válaszadási viselkedését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha hiányzik, az *objektum* alapértelmezetten az éppen fókuszban lévő **Form** objektum lesz.
- A tulajdonság beállításához használható *meghajtó* argumentum a felhasználó által előidézett eseményre való válaszadási viselkedését meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True	Az *objektum* eseményre történő reagálása megengedett. (Alapértelmezett.)
False	Az *objektum* eseményre történő reagálása tiltott.

Az **Enabled** tulajdonsággal az alkalmazás aktuális állapotában használni nem kívánt objektumát vagy a csak megjeleníteni kívánt vezérlőelemet futásidőben nem elérhetővé tehetjük, majd szükség esetén engedélyezhetjük. Az **Enabled** tulajdonság **False**

beállításával a **Timer** vezérlőelem belső időszámlálójának visszaszámlálása leáll, azonban az ezt követő engedélyezésnél a visszaszámlálás az **Interval** tulajdonságban adott értéktől indul ismét.

A **Menu** vezérlőelem esetén az **Enabled** tulajdonság rendes körülmények között írható és olvasható is, azonban az olyan menüknél csak olvasható, amelyeket a Visual Basic jelenít meg vagy ad át, mint például az Add-Ins menü Add-In Manager parancsa.

EOF

Szintaxis: *objektum*.**EOF**

Az **EOF** tulajdonság **Recordset** (valamint az elavult **Dynaset**, **Snapshot** és **Table**) objektumon érvényes, és az aktuális rekordnak egy objektum utolsó rekordja utáni pozícióját jelző értéket ad vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság által visszaadott, az aktuális rekord mutatójától függő **Boolean** érték értelmezése a következő:

True Az érvényes rekord pozíciója az utolsó rekord után van.
False A rekord az utolsó vagy azt megelőző bármelyik pozícióban van.

Az **EOF** (és **BOF**) tulajdonságot annak meghatározására lehet használni, hogy egy **Recordset** objektum tartalmaz-e rekordot, illetve hogy egy **Recordset** objektumban rekordról rekordra lépve túlléptük-e annak határait.

Ha rekordot nem tartalmazó **Recordset** objektumot nyitunk meg, az **EOF** (és **BOF**) tulajdonság értéke **True**, a rekordhalmaz **RecordCount** tulajdonsága pedig 0 lesz. Ha legalább egy rekordot tartalmazó **Recordset** van megnyitva, az aktuális rekord a halmaz első rekordja lesz, az **EOF** tulajdonság értéke pedig mindaddig **False** marad, ameddig a **MoveNext** metódussal az utolsó rekord utáni pozícióra nem akarunk lépni. Ilyen esetben nem lesz aktuális rekord sem kijelölve.

A **BOF** és **EOF** tulajdonságok különböző kombinációi esetén a megengedett **Move** metódusokat a következő táblázat foglalja össze.

Ha BOF és EOF	MoveFirst, MoveLast	MovePrevious, Move <0	Move 0	MoveNext, Move > 0
BOF =True, EOF = False	Megengedett	Hibát okoz	Hibát okoz	Megengedett
BOF =False, EOF = True	Megengedett	Megengedett	Hibát okoz	Hibát okoz
BOF =True, EOF = True	Hibát okoz	Hibát okoz	Hibát okoz	Hibát okoz
BOF =False, EOF = False	Megengedett	Megengedett	Megengedett	Megengedett

Ha egy rekordot tartalmazó **Recordset** objektumon a halmaz utolsó rekordját aktuális rekorddá tevő **MoveLast** metódus után a **MoveNext** metódust használjuk, az aktuális rekord érvénytelenné válik és az **EOF** értéke **True** lesz. Ha ilyenkor ismét használjuk a **MoveNext** metódust, hiba következik be.

A táblázatban megengedettnek jelölt metódusok használata nem okoz hibajelzést, de nem szükségszerűen eredményezi rekord sikeres kijelölését. A rekordot nem kijelölő **Move** esetében az **EOF** tulajdonság értéke a következő lesz:

	MoveFirst, MoveLast	**MovePrevious, Move <**	**Move 0**	**MoveNext, Move > 0**
Az EOF értéke	True	Változatlan	Változatlan	True

Az **EOF** tulajdonság értékét még a következő metódusok is módosíthatják:

- Mivel az **OpenRecordset** metódus egy **MoveFirst** metódust is meghív, így üres rekordhalmazon az **EOF** tulajdonságot **True** értékre állítja be.
- Minden sikeresen rekordot meghatározó **Move** metódus az **EOF** tulajdonságot **False** értékre állítja be.
- A **Delete** metódus, — még akkor is, ha a rekordhalmaz utolsó rekordját törli — változatlanul hagyja az **EOF** tulajdonság értékét.

EOFAction

Szintaxis: *objektum*.**EOFAction** [= *érték*]

Az **EOFAction** tulajdonság **Data** vezérlőelemen érvényes, és az **EOF** tulajdonság **True** értéke esetén bekövetkező műveletet megadó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a műveletet meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	A művelet:
vbEOFActionMoveLast	0	**MoveLast** — Az utolsó rekord marad az aktuális rekord. (Alapértelmezett.)
vbEOFAActionEOF	1	**EOF** — A **Recordset** végének elhagyása a **Data** vezérlőelem **Validate** eseményét okozza az utolsó rekordon, amit egy **Reposition** esemény követ az érvénytelenné vált (**EOF**) rekordon. Ekkor a **Data** vezérlőelem Move Next gombja nem el-

érhetővé válik.

vbEOFAActionAddNew	2	**AddNew** — Az utolsó rekord elhagyása a **Data** vezérlőelem **Validate** eseményét okozza az aktuális rekordon, amit egy automatikus **AddNew** esemény és egy **Reposition** esemény követ az új rekordon.

Ha az **EOFAction** tulajdonság értéke 2 vagy **vbEOFAActionAddNew** (**AddNew**), az aktuális rekord mutatójának a felhasználó által a **Data** vezérlőelem használatával **EOF**-ra változtatásakor az aktuális rekord a másolatpuffer egy új rekordjára pozícionálódik, és az újonnan csatolt rekord szerkeszthetővé válik. A rekord módosítása után az aktuális rekord mutatóját a **Data** vezérlőelem használatával elmozgatva a rekord automatikusan a **Recordset** rekordhalmazhoz adódik. Ha a rekordot nem módosítottuk a rekordmutató elmozgatása előtt, az új rekord törlődik. Ha a **Data** vezérlőelem használatával az aktuális rekordra állásnál másik rekordra állunk, egy másik új rekord jön létre.

A **Data** vezérlőelem használatával létrehozott **Recordset** rekordhalmaz programkódból történő kezelése esetén az **EOFAction** tulajdonság értékének nincs jelentősége, az csak az egérrel történő manipuláció mellett hatásos.

Az olyan esetekben, amikor a **Data** vezérlőelem **Recordset** rekordhalmaza nem ad vissza rekordot vagy az utolsó rekord is törlésre került, az *érték* argumentumban a 2 vagy **vbEOFAActionAddNew** érték használatával a programkód jelentősen egyszerűbbé tehető, mivel az új rekord aktuális rekordként mindig szerkeszthető lesz.

Exclusive

Szintaxis: *objektum*.**Exclusive** [= *érték*]

A tulajdonság **Data** vezérlőelemen érvényes, és a vezérlőelem alapjául szolgáló adatbázis egy- vagy többfelhasználós hozzáférésre történő megnyitását meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a felhasználói hozzáférési módot meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True	Az adatbázis csak egy felhasználó számára van megnyitva, és amíg be nem zárjuk, más felhasználó nem tudja elérni.
False	Az adatbázis több felhasználós használatra megnyitott, így azt mások is megnyithatják és az adataihoz hozzáférhetnek. (Alapértelmezett.)

A tulajdonság értéke a **DatabaseName**, **ReadOnly** és **Connect** tulajdonságokkal

együtt egy adatbázis megnyitására használható. (A Visual Basic Professional Edition esetén a tulajdonság az **OpenDatabase** metódus *kizárólagos* argumentumának felel meg.)

Az **Exclusive** tulajdonság csak **Database** objektum megnyitásához használható. A tulajdonság értékének futásidőben történő módosításakor elsőként a **Refresh** metódussal a változást érvényesíteni kell. Egy más által már megnyitott objektum kizárólagos hozzáférésre történő megnyitásának kísérlete kezelhető hibát okoz.

A kizárólagos hozzáférésre megnyitott **Database** gyorsabb működésű, és azt az alkalmazás akárhány példányban megnyithatja, azonban más, a rendszerünkben futó alkalmazás nem.

Az ODBC-n keresztül hozzáférhető adatbázisok figyelmen kívül hagyják az **Exclusive** tulajdonság beállítását.

EXEName

Szintaxis: *objektum*.**EXEName**

A tulajdonság **App** objektumon érvényes, és az éppen futó .EXE fájl kiterjesztés nélküli nevét, a programfejlesztési környezetben a projekt nevét adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Fields

Szintaxis:

Ez az **Index** objektumon használható, és egy indexben a kulcsmezőként érvényes **TableDef** mezőket jelző érték kiolvasására vagy beállítására használható tulajdonság a Visual Basic 4.0-s verziójában csak kompatibilitási okokból maradt meg. Helyette a **Field** objektumot javasolt használni egy **Index** objektum **Indexes** halmazában. Ebben az esetben minden egyes kulcsmezőhöz egy **Field** objektumot hozunk létre és az indexkulcsok jelölésére egy **Index** objektum **Fields** objektumgyűjteményéhez kapcsoljuk azokat.

FileDescription

Szintaxis: *objektum*.**FileDescription**

A tulajdonság **App** objektumon futásidőben csak olvasható és egy futó alkalmazás fájlleíró információs karakterláncát adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság beállítása a programfejlesztési fázisban az EXE Options párbeszédpanelen lehetséges.

FileName

Szintaxis: *objektum*.**Exclusive** [= *érték*]

A tulajdonság **ProjectTemplate** objektumon, valamint **CommonDialog** és **FileListBox** vezérlőelemen érvényes, és egy kiválasztott fájl elérési útjának és nevének kiolvasására vagy beállítására használható. A programfejlesztési fázisban csak a **CommonDialog** vezérlőelem számára elérhető. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, az elérési utat és fájlnevet tartalmazó karakteres kifejezés.

A futásidőben létrehozott vezérlőelem **FileName** tulajdonság értéke üres karakterlánc ("") lesz, jelezve, hogy nincs kiválasztott fájl.

A kezdeti fájlnevet a **CommonDialog** vezérlőelem **FileName** tulajdonságba egy párbeszédpanel megnyitása előtt lehet beállítani.

A tulajdonság értéke kiolvasáskor a lista aktuális fájlneve, kiválasztott elem hiányában üres karakterlánc lesz. A tulajdonság a **List(ListIndex)** értékével egyező értékű. Az elérési utat különállóan a **Path** tulajdonsággal lehet meghatározni.

A tulajdonság beállításnál a következőkre kell figyelemmel lenni:

- A tulajdonság beállításakor az *érték* argumentumban megadott meghajtó, elérési út vagy névminta automatikusan módosítja a **Drive**, **Path** és **Pattern** tulajdonságokat is.
- Az *érték* argumentumként létező fájlnevet tartalmazó karakterlánc a fájlt kijelöltté teszi.
- Az *érték* argumentumban az érvényes meghajtó módosításakor egy **PathChange**, a minta módosításakor egy **PatternChange**, egy létező fájlnév módosításakor pedig egy **DblClick** esemény következik be.
- A tulajdonság beállítása egy hálózati elérési út és fájlnév is lehet, például a `\\kiszolgalonev\megosztasnev\eleresi_ut` alakban.

FileNumber

Szintaxis:

Ez az **OLE** befogadó vezérlőelemeken érvényes, egy objektum mentéséhez vagy betöltéséhez szükséges fájlszám kiolvasására vagy beállítására, továbbá az utolsóként használt fájlnév kiolvasására használható tulajdonság a Visual Basic 4.0-s verziójában csak kompatibilitási okokból maradt meg. Helyette a **SaveToFile** és **ReadFromFile** metódus használata javasolt.

FileTitle

Szintaxis: *objektum*.**FileTitle**

A tulajdonság **CommonDialog** objektumon érvényes, és a megnyitandó vagy mentendő fájl (elérési út nélküli) nevet adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A felhasználó által a File párbeszédpanelen a megnyitandó vagy mentendő fájl kiválasztása, majd az **OK** gomb lenyomása után (hacsak a **cdlNoValidate** jelző nem beállított) a tulajdonság a kiválasztott fájl megnyitásához vagy mentéséhez használható értéket ad vissza.

FillColor

Szintaxis: *objektum*.**FillColor** [= *érték*]

A tulajdonság **Form** és **Printer** objektumon, valamint **PictureBox** és **Shape** vezérlőelemen érvényes, és a zárt körvonalú, így például a **Circle** és a **Line** metódusokkal készített kör és téglalap alakzatok belsejét kitöltő kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, az alakzat kitöltéséhez érvényes színt meghatározó konstans vagy színkód értéke.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
Normál RGB szín	A Color paletta vagy a programkódban az **RGB( )** vagy **QBColor( )** függvénnyel meghatározott szín.
Alapértelmezett szín	A Visual Basic objektumtárban a rendszer színkonstansával meghatározott szín. A Windows környezet a felhasználó színválasztását a Control Panel érvényes színével helyettesíti.

A **FillColor** tulajdonság alapértelmezett értéke 0, azaz a kitöltőszín fekete. A tulajdonság — a **Form** objektum kivételével — a **FillStyle** alapértelmezett beállítása esetén (*Transparent*) hatástalan.

FillStyle

Szintaxis: *objektum*.**FillStyle** [= *érték*]

A tulajdonság **Form** és **Printer** objektumon, valamint **Grid**, **PictureBox** és **Shape** vezérlőelemen érvényes: a **Grid** vezérlőelemen a vezérlőelem **Text** tulajdonságának

olyan beállítását meghatározó érték kiolvasására vagy beállítására használható, amely az összes kijelölt cellában megjeleníti az értékeket, a többi objektumon és vezérlőelemen pedig a zárt körvonalú, így például a **Circle** és a **Line** metódusokkal készített kör és téglalap alakzatok belsejét kitöltő minta kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, az alakzat kitöltéséhez érvényes mintát meghatározó **Integer** érték.

A **Grid** vezérlőelemen az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
0	*Single* — A **Text** tulajdonság megváltozása csak az aktív cellára hat.
1	*Repeat* — A **Text** tulajdonság változása mindegyik kiválasztott cellára hat.

A **Form** és **Printer** objektumon, valamint **PictureBox** és **Shape** vezérlőelemen az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
0	*Solid* — Egyszínű, homogén felület.
1	*Transparent* — Átlátszó felület. (Alapértelmezett.)
2	*Horizontal Line* — Vízszintes vonalkázás.
3	*Vertical Line* — Függőleges vonalkázás.
4	*Upward Diagonal* — Felfelé átlós vonalkázás.
5	*Downward Line* — Lefelé átlós vonalkázás.
6	*Cross* — Kockás felület.
7	*Diagonal Cross* — Átlósan kockás felület.

Ha a **FillStyle** tulajdonság értéke 1 (*Transparent*), a **FillColor** tulajdonság beállítása — a **Form** objektum kivételével — hatástalan.

Filter

Szintaxis 1: *objektum*.**Filter** [= *meghatározás1* | *szűrő1* | *meghatározás2* | *szűrő2* ...]

Szintaxis 2: *objektum*. **Filter** [= *érték*]

A **Filter** tulajdonság két különböző szintaxisa eltérő elemeken érvényes, és természetszerűleg eltérő hatású.

A Szintaxis 1 szerinti **Filter** tulajdonság **CommonDialog** vezérlőelemen érvényes, és egy párbeszédpanel Type listamezőjében érvényes szűrő kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentu-

mokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *meghatározás* argumentum a fájl típusát leíró karakteres kifejezés.
- A *szűrő* argumentum a fájlkiterjesztést megadó karakteres kifejezés.

A felhasználó a szűrővel egy párbeszédpanel fájllistájában megjelenő fájlok típusát tudja előírni, például a például szöveg és képfájl esetén a következő módon:

```
Text(*.txt)|*.txt|Pictures(*.bmp;*.ico)|*.bmp;*.ico
```

A *meghatározás* és *szűrő* argumentumok kizárólag a "|" karakterrel (ASCII 124) választhatók el egymástól, mégpedig úgy, hogy a "|" karakter előtt és után nem szabad szóközt tenni, különben a szóköz a megfelelő argumentum részévé válik.

Egy párbeszédpanelen több szűrő esetén az alapértelmezettet közülük a **FilterIndex** tulajdonsággal jelölhetjük ki.

A Szintaxis 2 szerinti **Filter** tulajdonság eredményhalmaz és pillanatfelvétel típusú **Recordset** (valamint az elavult **Dynaset** és **Snapshot**) objektumokon érvényes, és egy következőleg megnyitott **Recordset** objektumbeli rekordokat meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **String** típusú karakteres kifejezés, amely egy SQL utasítás **WHERE** záradékát a **WHERE** kulcsszó nélkül tartalmazza, például a `"[ Region ] = 'NY' "`alakban.

A **Filter** tulajdonsággal egy létező **Recordset** objektumon alapuló új **Recordset** megnyitásakor az objektumból visszaadott rekordok mennyiségét lehet csökkenteni, így gyorsabbá téve az objektum megnyitását.

Ha dátum szerinti szűrésre van szükség, még akkor is US dátumalakot (hónap-nap-év) kell megadni, ha nem az US verzió szerinti Microsoft Jet adatbázis-szerkesztőt használjuk, különben a dátumszűrő nem fog megfelelően működni.

FilterIndex

Szintaxis: *objektum*.**FilterIndex** [= *érték*]

A tulajdonság **CommonDialog** vezérlőelemen érvényes, és egy Open vagy Save As párbeszédpanelen a **Filter** tulajdonságban adott listán alapértelmezettként érvényes szűrő kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az alapértelmezett

szűrőt megadó numerikus kifejezés. Az első listaelem a **Filter** = 1 beállításnak felel meg.

FirstRow

Szintaxis: *objektum*.**FirstRow** [= *érték*]

A tulajdonság **DBGrid** vezérlőelemen érvényes, és a vezérlőelem első látható sorához tartozó könyvjelzőt tartalmazó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a vezérlőelem első látható sorához tartozó könyvjelzőt megadó karakteres kifejezés.

A **FirstRow** tulajdonság beállítása a **DBGrid** vezérlőelem olyan görgetését eredményezi, amelynek eredményeként az adott sor a vezérlőelem legfelső sorává válik.

FixedAlignment

Szintaxis: *objektum*.**FixedAlignment**(*oszlop*) [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Grid** vezérlőelemen érvényes, és egy oszlop rögzített cellájában az adat helyzetét meghatározó érték kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *oszlop* argumentum egy oszlop száma.
- Az *érték* argumentum az adat helyzetét meghatározó **Integer** érték.

Az *érték* lehetséges beállításai és azok értelmezése a következő:

Ha az *érték*:	Értelmezése:
0	Az adatok balra igazítottak.(Alapértelmezett.)
1	Az adatok jobbra igazítottak.
2	Az adatok középre igazítottak.
3	A **ColAlignment** beállításának megfelelő helyzet.

A **FixedAlignment** a **ColAlignment** tulajdonsághoz hasonló funkciójú, azzal az eltéréssel, hogy a **FixedAlignment** csak a rögzített cellák beállítására hat. Használata például akkor előnyös, ha egy fejlécet az oszlop többi elemétől eltérően kívánunk pozícionálni.

FixedCols

Szintaxis: *objektum*.**FirstCols** [= *érték*]

A tulajdonság **Grid** vezérlőelemen érvényes, és a vezérlőelembeli rögzített oszlopok számának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a vezérlőelem összes rögzített oszlopainak számát megadó numerikus kifejezés.

Alapértelmezetten a **Grid** egy rögzített oszlopot tartalmaz, amely a vezérlőelem bal oldali, állandó tartalmú adatoszlopa. A vezérlőelem nem kötelezően tartalmaz rögzített oszlopot, másrészről pedig nem lehet minden oszlopa rögzített, legalább egy nem rögzített oszlopának is kell lennie. A rögzített oszlopot a többitől a képernyőn szürke, nem változtatható színe is megkülönbözteti. A nem rögzített mezők görgetésekor a rögzített mező mindig helybenmarad.

A rögzített mezők elsősorban számolótáblákon használatosak a táblázat sorait azonosító név, sorszám vagy más jelzés tárolására.

FixedRows

Szintaxis: *objektum*.**FirstRows** [= *érték*]

A tulajdonság **Grid** vezérlőelemen érvényes, és a vezérlőelembeli rögzített sorok számának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a vezérlőelem összes rögzített sorainak számát megadó numerikus kifejezés.

A **FixedRows** tulajdonságra vonatkozó további tudnivalók értelemszerűen azonosak a **FixedCols** tulajdonságnál leírtakkal.

Flags

Szintaxis: *objektum*.**Flags** [= *érték*]

A tulajdonság **CommonDialog** vezérlőelemen érvényes, és egy Color, File (Open és Save As), Font vagy Print párbeszédpanel beállításainak megadására vagy kiolvasására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

- A tulajdonság beállításához használható *érték* argumentum egy vagy több, a párbeszédpanelek beállításaihoz érvényes szimbolikus konstans vagy **Integer** típusú érték.

A különböző párbeszédpaneleken érvényes beállítások a következők:

A Color párbeszédpanelen:

Ha a szimbolikus konstans:	Értéke:	Értelmezése:
cdlFullOpen	&H2&	A panel létrehozásakor a felhasználó által definiált szín létrehozását lehetővé tévő (Define Custom Color) részt is magában foglaló párbeszédpanelt jelenít meg. Ha a jelző nincs beállítva, a felhasználónak a saját szín megadásához a Define Custom Color gombot kell kiválasztania.
cdlPreventFullOpen	&H4&	Nem elérhetővé teszi a Define Custom Color parancsgombot, így a felhasználó nem definiálhat saját színeket.
cdlRGBIni	&H1&	A párbeszédpanelen az inicializálási színeket állítja be.
cdlShowHelp	H8&	A párbeszédpanelen egy Help gombot is elhelyez.

A File (Open és Save As) párbeszédpanelen:

Ha a szimbolikus konstans:	Értéke:	Értelmezése:
cdlAllowMultiselect	&H200&	Futásidőben megengedett több fájl egyidejű kijelölése is a File Name listamezőben a SHIFT és a LE vagy FEL billentyűk egyidejű használatával. Ezt követően a **FileName** tulajdonság az összes kiválasztott fájl nevét tartalmazó, a neveket nullával elválasztó karakterláncot ad vissza.
cdlCreatePrompt	&H2000&	A felhasználó számára új, még nem létező fájl létrehozására felszólító párbeszédpanelt jelenít meg. A tulajdonság a **cdlPathMustExist** és **cdlFileMustExist** jelzőket automatikusan beállítja.
cdlExtensionDifferent	&H400&	A visszaadott fájlnév kiterjesztése eltér a **DefaultExt** tulajdonságban meghatározottól. Ha a **DefaultExt** tulajdonság **Null**, a kiterjesztés egyezik a mintával, vagy ha a fájlnak nincs kiterjesztése, a jelző nem lesz beállított. A jelző értéke a párbeszédpanel bezárásakor ellenőrizhető.

cdlFileMustExist	&H1000&	A felhasználó a File Name párbeszédpanelen csak meglévő fájl nevét adhatja meg. Ha a jelző beállított és a felhasználó érvénytelen fájlnevet ad meg, figyelmeztető üzenet jelenik meg. A tulajdonság automatikusan beállítja a **cdlPathMustExist** jelzőt.
cdlHideReadOnly	&H4&	A Read Only választómező rejtetté válik.
cdlNoChangeDir	&H8&	A párbeszédpanel megnyitásakor érvényes könyvtár válik aktuális könyvtárrá.
cdlNoReadOnlyReturn	&H8000&	A visszaadott fájl csak olvasható attribútuma nem lesz beállított és nem lesz írásvédett könyvtárban.
cdlNoValidate	&H100&	A párbeszédpanel a visszaadott fájlnévben érvénytelen karaktereket is megenged.
cdlOverwritePrompt	&H2&	A kiválasztott fájlnév létezése esetén a Save As párbeszédpanelen figyelmeztető üzenetmezőt hoz létre, amelyen a felhasználó engedélyezheti a fájl felülírását.
cdlPathMustExist	&H800&	A felhasználó csak érvényes elérési utat adhat meg. Ha a jelző beállított és a felhasználó érvénytelen elérési utat ad meg, figyelmeztető üzenet jelenik meg.
cdlReadOnly	&H1&	A párbeszédpanel létrehozásakor a Read Only jelölőnégyzetet kiválasztottá teszi, a párbeszédpanel bezárása után pedig a jelölőnégyzet állapotát jelzi.
cdlShareAware	&H4000&	A megosztási hibák figyelmen kívül maradnak.
cdlShowHelp	&H10&	A párbeszédpanelen egy Help gombot is elhelyez.

A Font párbeszédpanelen:

Ha a szimbolikus konstans:	Értéke:	Értelmezése:
cdlANSIOnly	&H400&	Csak egy Windows karaktertípus kiválasztása megengedett. Ha a jelző beállított, a csak szimbólumokat tartalmazó karaktertípusok nem választhatók ki.
cdlBoth	&H3&	A párbeszédpanel a nyomtatón és képernyőn használható karaktertípusok listáját jeleníti meg. A nyomtatóra vonatkozó eszközkapcsolatot a **hDC** tulajdonság jelöli ki.

cdlEffects	&H100&	Csak áthúzási-, aláhúzási- és színhatások megengedettek.
cdlFixedPitchOnly	&H4000&	Csak rögzített méretű karaktertípusok kiválasztása megengedett.
cdlForceFontExist	&H10000&	Hibaüzenetet jelenít meg, ha a felhasználó nem létező karaktertípust vagy stílust kísérel meg kiválasztani.
cdlLimitSize	&H2000&	Csak a **Min** és **Max** tulajdonságban adott határok közötti méretű karakternagyság választható ki.
cdlNoSimulations	&H1000&	A GDI (grafikus eszközinterfész) karaktertípusok szimulációja nem megengedett.
cdlNoVectorFonts	&H800&	A vektorizált karaktertípusok kiválasztása nem megengedett.
cdlPrinterFonts	&H2&	A párbeszédpanelen csak a **hDC** tulajdonságban adott, a nyomtató által támogatott karaktertípusok listája jelenik meg.
cdlScalableOnly	&H20000&	Csak méretezhető karaktertípusok kiválasztása megengedett.
cdlScreenFonts	&H1&	A párbeszédpanelen csak a rendszer által támogatott képernyő-karakterkészlet jelenik meg.
cdlShowHelp	&H4&	A párbeszédűrlapon egy Help gombot is elhelyez.
cdlTTOnly	&H40000&	Csak *TrueType* karaktertípusok kiválasztása megengedett.
cdlWYSIWYG	&H8000&	Csak a nyomtatón és képernyőn is használható karaktertípusok kiválasztása engedélyezett. A jelző beállítása automatikusan beállítottá teszi a **cdlBoth** és **cdlScalableOnly** jelzőket is.

A **Font** párbeszédpanel beállítása előtt a **Flags** tulajdonságot **cdlScreenFonts**, **cdlPrinterFonts** vagy **cdlBoth** értékre kell állítani, különben `No Fonts Exist` hibajelzést kapunk.

A Print párbeszédpanelen:

Ha a szimbolikus konstans:	Értéke:	Értelmezése:
cdlAllPages	&H0&	Beállítja vagy kiolvassa az All Pages választógomb állapotát.
cdlCollate	&H10&	Beállítja vagy kiolvassa a Collate jelölőnégyzet állapotát.

cdlDisablePrintTo File	&H80000&	Nem elérhetővé teszi a Print to File jelölőnégyzetet.
cdlHidePrintToFile	&H100000&	Rejtetté teszi a Print to File jelölőnégyzetet.
cdlNoPageNums	&H8&	Nem elérhetővé teszi a Pages választógombot és a kapcsolódó szövegbeviteli vezérlőelemeket.
cdlNoSelection	&H4&	Nem elérhetővé teszi a Selection választógombot.
cdlNoWarning	&H80&	Letiltja az alapértelmezett nyomtató hiányára figyelmeztető üzenet megjelenítését.
cdlPageNums	&H2&	Beállítja vagy kiolvassa a Pages választógomb állapotát.
cdlPrintSetup	&H40&	A Print helyett a Print Setup párbeszédpanelt jeleníti meg.
cdlPrintToFile	&H20&	Beállítja vagy kiolvassa a Print to File jelölőnégyzetet állapotát.
cdlReturnCD	&H100&	A párbeszédpanelen kiválasztott nyomtató eszközkapcsolatát (a párbeszédűrlap **hDC** tulajdonságát) adja vissza.
cdlReturnIC	&H200&	A párbeszédpanelen kiválasztott nyomtató információkapcsolatát (a párbeszédpanel **hDC** tulajdonságát) adja vissza. Az információkapcsolat eszközkapcsolat létrehozása nélkül ad lehetőséget az eszközről való információszerzésre.
cdlSelection	&H1&	Beállítja vagy kiolvassa a Selection választógombot. Ha a jelzők közül sem a **cdlPageNums**, sem a **cdlSelection** sincs megadva, az All választógomb kiválasztott állapotba kerül.
cdlShowHelp	&H800&	A párbeszédpanelen egy Help gombot is megjelenít.
cdlUseDevModeCopies	&H40000&	Ha a nyomtató nem támogatja a többszörös másolatok készítését, a jelző a másolatszerkesztési vezérlőelemet nem elérhetővé teszi. Ha a nyomtatóvezérlés nem támogatja a többszörös másolatok készítését, a jelző beállítása eredményeként a párbeszédpanel a szükséges másolatok számát a **Copies** tulajdonságban fogja tárolni.

A táblázatban szereplő szimbolikus konstansok listája az Object Browser Micro-

soft CommonDialog Control (MsComDlg) objektumtárában található. Az indító űrlap deklarációs részében a **Const** kulcsszóval további jelzők is definiálhatók.

Egyidőben több jelzőt azok **Or** művelettel (**&H10& Or &H200&**) történő összekapcsolásával vagy a jelzők értékeinek összegként (**&H210&**) lehet megadni.

FontBold

Szintaxis: *objektum*.**FontBold** [= *érték*]

A tulajdonság **CheckBox**, **ComboBox**, **CommandButton**, **CommonDialog**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **Label**, **ListBox**, **OptionButton**, **PictureBox** és **TextBox** vezérlőelemeken, valamint **Form** és **Printer** objektumon érvényes, és az objektumban a félkövér karakterstílus kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a félkövér karakterstílust meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True A karakterek megjelenése félkövér. A **CommonDialog** vezérlőelem kivételével alapértelmezett beállítás.

False A karakterek megjelenése nem félkövér. A **CommonDialog** vezérlőelemen alapértelmezett beállítás.

Ezzel a tulajdonsággal az objektumok és vezérlőelemek szövegének megjelenése állítható be, a programfejlesztési fázisban a Properties űrlapon, a futásidőben pedig a programkódból.

A **Form** és **Printer** objektumok, valamint a **PictureBox** vezérlőelem esetén a **FontBold** tulajdonság beállítása nem módosítja az objektumon és vezérlőelemen már megjelent grafikát vagy szöveget, míg az egyéb vezérlőelemeknél a tulajdonság beállítása azonnal hat a képernyőtartalomra.

A tulajdonság **CommonDialog** párbeszédpanellel együtt történő használatához a párbeszédpanel **Flags** tulajdonságának beállítva kell lennie.

A Visual Basicben használható karaktertípusok a rendszer összeállításától, a monitor és a nyomtató jellemzőitől függők. A karaktertípusokkal kapcsolatos tulajdonságoknak az adott típus számára megengedett beállításúaknak kell lenniük.

A **FontBold** (valamint **FontItalic**, **FontStrikethru FontUnderline** és **FontSize**) tulajdonság beállítását a **FontName** tulajdonság beállítása előtt kell végrehajtani.

FontCount

Szintaxis: *objektum*.**FontCount**

Ez a **Printer** és **Screen** objektumokon érvényes, a programfejlesztési fázisban nem használható, futásidőben pedig csak olvasható tulajdonság az éppen érvényes megjelenítő eszközön (monitoron vagy aktív nyomtatón) használható karaktertípusok számát adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A rendelkezésre álló karaktertípusok a **FontCount** és a **Fonts** tulajdonságok együttes használatával listázhatók ki.

A Visual Basicben használható karaktertípusok a rendszer konfigurációjától és a hozzá csatlakoztatott megjelenítő eszközöktől függően eltérők lehetnek.

FontItalic

Szintaxis: *objektum*.**FontItalic** [= *érték*]

A tulajdonság **CheckBox**, **ComboBox**, **CommandButton**, **CommonDialog**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **Label**, **ListBox**, **OptionButton**, **PictureBox** és **TextBox** vezérlőelemeken, valamint **Form** és **Printer** objektumon érvényes, és az objektumban a döntött karakterstílus kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a döntött karakterstílust meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True A karakterek megjelenése döntött.
False A karakterek megjelenése nem döntött. (Alapértelmezett.)

A tulajdonság használatával kapcsolatos további tudnivalók a **FontBold** tulajdonságnál már leírtakkal értelemszerűen azonosak.

FontName

Szintaxis: *objektum*.**FontName** [= *karaktertípus*]

A tulajdonság **CommonDialog** vezérlőelemen érvényes, és az objektumban szöveg megjelenítésére vagy futásidejű rajzolási vagy nyomtatási műveletben használt karaktertípus nevének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *karaktertípus* argumentum a használt karaktertípus nevét megadó karakteres kifejezés.

A **FontName** tulajdonság alapértelmezett értékét az operációs rendszer határozza meg, a Visual Basicben rendelkezésre álló karaktertípusok pedig a rendszer konfigurációjától és a hozzá csatlakoztatott megjelenítő eszközöktől (monitor és nyomtató) függően eltérők lehetnek. Természetszerűleg a vonatkozó tulajdonságok csak az érvényes karaktertípusokra értelmezettek.

A **FontName** tulajdonság beállítását a kapcsolódó **FontBold**, **FontItalic**, **FontSize**, **FontStrikethru** és **FontUnderline** tulajdonságok beállítása előtt kell végrehajtani.

A futásidőben a rendszer által használható karaktertípusokról a **FontCount** és **Fonts** tulajdonságok használatával lehet informálódni.

Meg kell jegyezni, hogy a **FontName** tulajdonság a Visual Basic 4.0-s verziójában **CheckBox**, **ComboBox**, **CommandButton**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **Label**, **ListBox**, **OptionButton**, **PictureBox** és **TextBox** valamint **Form** és **Printer** objektumon kompatibilitási okokból továbbra is használható maradt, azonban helyette az új **Font** objektumtulajdonságok alkalmazása javasolt.

Fonts

Szintaxis: *objektum*.**Fonts**(*index*)

Ez a **Printer** és **Screen** objektumokon érvényes tulajdonság az éppen érvényes megjelenítő eszközön (monitor és nyomtató) rendelkezésre álló összes karaktertípus nevét adja vissza. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum egy 0 és a **FontCount**-1 közötti **Integer** érték.

A **FontCount** és **Fonts** tulajdonságok együttesen a megjelenítő eszközökön (monitor és nyomtató) használható karaktertípusok listázására alkalmasak. A Visual Basicben rendelkezésre álló karaktertípusok a rendszer konfigurációjától és a hozzá csatlakoztatott megjelenítő eszközöktől függően eltérők lehetnek.

FontSize

Szintaxis: *objektum*.**FontSize** [= *érték*]

A tulajdonság **CommonDialog** vezérlőelemen érvényes, és az objektumban szöveg megjelenítésére vagy futásidejű rajzolási vagy nyomtatási műveletben használt karaktertípus méretének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a karaktertípus méretét pontban megadó numerikus kifejezés. A maximális érték 2160 pont lehet.

A **FontSize** tulajdonság szövegek méretszerinti formázására használható. Alapértelmezett értékét az operációs rendszer határozza meg, amelyet a **FontSize** tulajdonságon keresztül módosíthatunk.

A Visual Basicben rendelkezésre álló karaktertípusok a rendszer konfigurációjától és a hozzá csatlakoztatott megjelenítő eszközöktől (monitor és nyomtató) függően eltérők lehetnek. Természetszerűleg a vonatkozó tulajdonságok csak az érvényes karaktertípusokra értelmezettek.

A **FontSize** (valamint **FontBold**, **FontItalic**, **FontStrikethru** és **FontUnderline**) tulajdonság beállítását a **FontName** tulajdonság beállítása előtt kell végrehajtani. A 8 pontnál kisebb méretű *TrueType* karakterek esetén, a pontméret a **FontSize**, majd a **FontName**, végül ismét a **FontSize** tulajdonsággal állítható be, mivel a Windows a 8 pontnál kisebb *TrueType* karakterként eltérő karaktertípust használ.

Meg kell jegyezni, hogy a **FontSize** tulajdonság a Visual Basic 4.0-s verziójában **CheckBox**, **ComboBox**, **CommandButton**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **Label**, **ListBox**, **OptionButton**, **PictureBox** és **TextBox** valamint **Form** és **Printer** objektumon kompatibilitási okokból továbbra is használható maradt, azonban helyette az új **Font** objektumtulajdonságok alkalmazása javasolt.

FontStrikethru

Szintaxis: *objektum*.**FontStrikethru** [= *érték*]

A tulajdonság **CheckBox**, **ComboBox**, **CommandButton**, **CommonDialog**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **Label**, **ListBox**, **OptionButton**, **PictureBox** és **TextBox** vezérlőelemeken, valamint **Form** és **Printer** objektumon érvényes, és az objektumban egy vonallal áthúzott karakterstílus kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az áthúzott karakterstílust meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True	A karakterek megjelenése áthúzott.
False	A karakterek megjelenése nem áthúzott. (Alapértelmezett.)

A tulajdonság használatával kapcsolatos további tudnivalók a **FontBold** tulajdonságnál már leírtakkal értelemszerűen azonosak.

FontTransparent

Szintaxis: *objektum*.**FontTransparent** [= *érték*]

A tulajdonság **PictureBox** vezérlőelemen, valamint **Form** és **Printer** objektumon érvényes, és a háttérszövegnek vagy grafikának a karakterek körüli területen való megjelenését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a háttérszöveg és grafika megjelenési állapotát meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True A háttérszöveg vagy grafika átlátszik a karakterek körüli maszkon. (Alapértelmezett.)
False A szövegmaszk fedi a háttér szövegét vagy grafikáját.

Ezzel a tulajdonsággal az objektumok és vezérlőelemek szövegének megjelenése állítható be, a programfejlesztési fázisban a Properties űrlapon, a futásidőben pedig a programkódból. A **FontTransparent** tulajdonság beállítása nem módosítja az objektumon vagy vezérlőelemen már megjelent grafikát vagy szöveget.

FontUnderline

Szintaxis: *objektum*.**FontUnderline** [= *érték*]

A tulajdonság **CheckBox**, **ComboBox**, **CommandButton**, **CommonDialog**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **Label**, **ListBox**, **OptionButton**, **PictureBox** és **TextBox** vezérlőelemeken, valamint **Form** és **Printer** objektumon érvényes, és az objektumban aláhúzott karakterstílus kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az aláhúzott karakterstílust meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True A karakterek megjelenése aláhúzott.
False A karakterek megjelenése nem aláhúzott. (Alapértelmezett.)

A tulajdonság használatával kapcsolatos további tudnivalók a **FontBold** tulajdonságnál már leírtakkal értelemszerűen azonosak.

ForeColor

Szintaxis: *objektum*.**ForeColor** = [*érték*]

A **ForeColor** tulajdonság **CheckBox**, **ComboBox**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox** és **TextBox** vezérlőelemeken, valamint **Column**, **Form** és **Printer** objektumon érvényes, és az objektumban megjelenő szöveg és grafika előtérszínének kiolvasására vagy beállítására használható.

A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, az előtérszínt meghatározó színkód értéke.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
Normál RGB szín	A Color paletta vagy a programkódban az **RGB** vagy **QBColor** függvénnyel meghatározott szín.
Alapértelmezett szín	A Visual Basic objektumtárban a rendszer színkonstansával meghatározott szín. A Windows környezet a felhasználó színválasztását a Control Panel érvényes színével helyettesíti.

A programfejlesztési fázisban minden űrlapra és vezérlőelemre a **vbWindowText** konstans szerinti előtérszín lesz érvényes.

A **ForeColor** tulajdonság beállítása nem hat a már meglévő szövegre vagy grafikára.

A normál RGB színeket meghatározó *szín* számértékének érvényes tartománya 0 és 16777215 (**&HFFFFFF**) között lehet. A felső byte értéke ebben a tartományban 0, az alsó bitcsoport a legalacsonyabb bittől felfelé a vörös, zöld és kék színösszetevőt határozza meg. Ezen összetevők 0 és 255 (**&HFF**) közötti értékek lehetnek. Ha a felső byte nem nulla, a Visual Basic a Control Panel beállításának és az Object Browser objektumtárában található konstansoknak megfelelő rendszerszíneket használja.

Foreign

Szintaxis: *objektum*.**Foreign**

Ez az **Index** objektumokon érvényes, csak olvasható tulajdonság az objektum egy táblában idegen kulcskénti értelmezését jelző értéket ad vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság által visszaadott **Boolean** érték értelmezése a következő:

True Az **Index** objektum idegen kulcsot képvisel.
False Az **Index** objektum nem idegen kulcsot képvisel.

Az idegen kulcs egy idegen táblában egy vagy több mezőt tartalmaz, amely egyértelműen azonosítja egy elsődleges tábla mindegyik sorát.

A Microsoft Jet egy **Index** objektumot hoz létre az idegen táblához és a hivatkozási integritást erősítő viszonylat létrehozásakor beállítja a **Foreign** tulajdonságot.

ForeignName

Szintaxis: *objektum*.**ForeignName** = [*érték*]

A tulajdonság **Field** objektumon érvényes, és egy idegen táblában egy elsődleges tábla mezőjének megfelelő **Field** objektum nevét egy viszonylatban megadó érték kiolvasására vagy beállítására használható.

A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **TableDef** objektum **Fields** halmazában lévő **Field** objektum nevét tartalmazó **String** típusú kifejezés.

A **Relation** objektumbeli **Fields** objektumgyűjteményből csak egy **Field** objektum támogathatja a **ForeignName** tulajdonságot. Ha a **Relation** nincs is a **Database** objektumhoz kapcsolva, de a **Field** egy **Relation** objektumhoz kapcsolt, a **ForeignName** tulajdonság módosítható. Amint a **Relation** objektum egy adatbázishoz kapcsolódik, a **Field** objektum tulajdonságai csak olvashatókká válnak.

A **Relation** objektumnál a **Table** és **ForeignName** tulajdonságok *érték* beállítása egy viszonyban az elsődleges és az idegen táblákat határozza meg, a táblákkal kapcsolatban álló mezőket pedig a **Field** objektum **Name** és **ForeignName** tulajdonságainak *érték* beállítása adja meg.

ForeignTable

Szintaxis: *objektum*.**ForeignTable** = [*érték*]

A tulajdonság **Relation** objektumon érvényes, és egy viszonylatban egy idegen tábla nevét megadó érték kiolvasására vagy beállítására használható. A tulajdonság objektumgyűjteményhez még nem tartozó új **Relation** objektum esetén írható és olvasható, míg **Relations** objektumgyűjteményen belüli **Relation** objektum esetén csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

- A tulajdonság beállításához használható *érték* argumentum egy **Database** objektum **TableDefs** halmazában lévő tábla nevét tartalmazó **String** típusú kifejezés.

A **Relation** objektum **ForeignTable** tulajdonságának beállítása a **TableDef** vagy **QueryDef** objektum **Name** tulajdonságának idegen táblát vagy lekérdezést jelölő beállítása, a **Name** tulajdonság beállítása pedig a **TableDef** vagy **QueryDef** objektum **Name** tulajdonságának elsődleges táblát vagy lekérdezést jelölő beállítása.

Format

Szintaxis: *objektum*.**Format** [= *forma*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **OLE** befogadó vezérlőelemen használható, és az objektumot létrehozó alkalmazás számára küldött vagy abból fogadott adat formájának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **Data** és **DataText** tulajdonságokkal használt formát megadó karakteres kifejezés.

Egy adott objektumosztály által elfogadható formák listáját az **ObjectAcceptFormats**, **ObjectAcceptFormatsCount**, **ObjectGetFormats** és **ObjectGetFormats** tulajdonságokkal kaphatjuk meg.

Számos alkalmazási program csak egy vagy két adatformát — például a Microsoft Draw csak a CF_METAFILEPICT — formát támogat. Noha a CF_METAFILEPICT hasonlít a Visual Basic objektumtárának **vbCFMetafile** belső konstansához, azonban csak betűszerinti karakterláncként, az `Ole1.Format = "CF_METAFILEPICT"` alakban használható.

Több esetben az objektum által a szolgáltatott (**ObjectGetFormats**) és az elfogadható (**ObjectAcceptFormats**) formák eltérők lehetnek.

FromPage

Szintaxis: *objektum*.**FromPage** [= *érték*]

Ez a **CommonDialog** (Print Dialog) vezérlőelemen használható tulajdonság a From szövegbeviteli mező értékének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az első nyomtatandó oldal számát megadó numerikus kifejezés.

A **FromPage** tulajdonság csak a **Flags** tulajdonság **cflPageNums** jelzőjének beállított állapotában érvényes.

GridLines

Szintaxis: *objektum*.**GridLines** [= *érték*]

A tulajdonság **Grid** vezérlőelemen használható, és a cellák közötti elválasztóvonal láthatóságát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a rácsvonal láthatóságát meghatározó **Boolean** érték.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

True	A rácsvonalak láthatók. (Alapértelmezett.)
False	A rácsvonalak nem láthatók.

GridLineWidth

Szintaxis: *objektum*.**GridLineWidth** [= *érték*]

A tulajdonság **Grid** vezérlőelemen használható, és a cellák közötti elválasztóvonal vastagságát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a rácsvonal képpontban mért szélességét 1 (alapértelmezett) és 10 között meghatározó **Integer** érték.

Handle

Szintaxis: *objektum*.**Handle**

Ez a **Picture** objektumon érvényes tulajdonság az objektumbeli grafika kezelőjét adja vissza. A szintaxisban az *objektum* azonosítója egy objektumkijelölő objektumkifejezés. A tulajdonság által visszaadott érték a **Type** tulajdonság beállításától a következők szerint függ:

Ha a **Type** értéke:	A **Handle** értéke:
1 (Bitkép)	HBITMAP
2 (Metafájl)	HMETAFILE
3 (Ikon)	HICON vagy HCURSOR

A **Handle** tulajdonság a kezelő egy grafikának DLL-ben lévő függvény vagy Windows API hívás részeként történő átadásakor használatos.

hDC

Szintaxis: *objektum*.**hDC**

Ez a **Form** és **Printer** objektumon, valamint **PictureBox** vezérlőelemen érvényes tulajdonság a Microsoft Windows operációs környezet által egy objektum eszközkapcsolatához adott kezelőt adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az alkalmazásban a Microsoft Windows operációs környezet a rendszerkijelzést a **Printer** objektumhoz és minden egyes űrlaphoz és **PictureBox** vezérlőelemhez egy eszközkapcsolatot beállítva szervezi. Egy objektum eszközkapcsolatának kezelőjére a **hDC** tulajdonsággal lehet hivatkozni, amely a Windows API hívások számára egy értéket ad át.

A **CommonDialog** vezérlőelem esetén a **hDC** tulajdonság a Print párbeszédpanelen kiválasztott nyomtató számára eszközkapcsolatot ad vissza, ha a **Flags** tulajdonságban a **cdlReturnDC**, és információs kapcsolatot, ha a **cdlReturnIC** jelző van beállítva.

A **hDC** tulajdonság használatakor ne felejtsük el, hogy értéke a program futása alatt is módosulhat, ezért az értékének változóban való tárolása helyett minden egyes felhasználáskor kell kiolvasni. Például az **AutoRedraw** értéke hatással van a **hDC** tulajdonság értékére: ha egy **Form** objektum vagy **PictureBox** vezérlőelem **AutoRedraw** tulajdonsága **True**, a **hDC** tulajdonság — az **Image** tulajdonsággal azonos módon — az állandó háttér bitképes grafikájának eszközkapcsolat-kezelőjeként működik, ha **False**, a **hDC** a **Form** objektum vagy a **PictureBox** vezérlőelem **hDC** tulajdonságának aktuális értékét adja. A **hDC** tulajdonság beállítása futásidőben az **AutoRedraw** beállításától függetlenül is változhat.

HeadBackColor

Szintaxis: *objektum*.**HeadBackColor** [= *érték*]

A tulajdonság **Column** objektumon használható, és a **DBGrid** vezérlőelem kiválasztott oszlopának háttérszínét jelző érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, a háttérszínt meghatározó konstans vagy színkód értéke.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
Normál RGB szín	A Color paletta vagy a programkódban az **RGB** vagy **QBColor** függvénnyel meghatározott szín.
Alapértelmezett szín	A Visual Basic objektumtárban a rendszer színkonstansával meghatározott szín. Az alapértelmezett értéke **vbButtonFace**. A Windows környezet a felhasználó színválasztását a Control Panel érvényes színével helyettesíti.

A normál RGB színeket meghatározó *érték* számértékének érvényes tartománya 0 és 16 777 215 (**&HFFFFFF**) között lehet. A felső byte értéke ebben a tartományban 0, az alsó bitcsoport a legalacsonyabb bittől felfelé a vörös, zöld és kék színösszetevőt határozza meg. Ezen összetevők 0 és 255 (**&HFF**) közötti értékek lehetnek. Ha a felső byte nem nulla, a Visual Basic a Control Panel beállításának és az Object Browser objektumtárában található konstansoknak megfelelő rendszerszíneket használja.

HeadFont

Szintaxis: *objektum*.**HeadFont** [= *érték*]

A tulajdonság **DBGrid** vezérlőelemen használható, és a vezérlőelem oszlopfejlécében karaktertípust megadó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a karaktertípust meghatározó karakteres kifejezés. Az alapértelmezett érték az oszlop félkövér stílusú karaktertípusa.

A **HeadFont** tulajdonság módosítása a fejléc méretét automatikusan az új karaktertípusnak megfelelően változtatja meg.

HeadForeColor

Szintaxis: *objektum*.**HeadForeColor** [= *érték*]

A tulajdonság **DBGrid** objektumon használható, és a **DBGrid** vezérlőelem kiválasztott oszlopának előtérszínét jelző érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy, az előtérszínt meghatározó konstans vagy színkód értéke.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
Normál RGB szín	A Color paletta vagy a programkódban az **RGB** vagy **QBColor** függvénnyel meghatározott szín.
Alapértelmezett szín	A Visual Basic objektumtárban a rendszer színkonstansával meghatározott szín. Az alapértelmezett értéke **vbButtonFace**. A Windows környezet a felhasználó színválasztását a Control Panel érvényes színével helyettesíti.

A normál RGB színeket meghatározó *érték* számértékének érvényes tartománya 0 és 16 777 215 (**&HFFFFFF**) között lehet. A felső byte értéke ebben a tartományban 0, az alsó bitcsoport a legalacsonyabb bittől felfelé a vörös, zöld és kék színösszetevőt határozza meg. Ezen összetevők 0 és 255 (**&HFF**) közötti értékek lehetnek. Ha a felső byte nem nulla, a Visual Basic a Control Panel beállításának és az Object Browser objektumtárában található konstansoknak megfelelő rendszerszíneket használja.

HeadLines

Szintaxis: *objektum*.**HeadLines** [= *érték*]

A tulajdonság **DBGrid** vezérlőelemen használható, és a vezérlőelem oszlopfejlécében megjelenő szöveg sorainak számát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a fejlécbeli sorok számát megadó, 0 és 10 közötti, **Single** típusú érték. Az *érték* = 0 beállítás eltávolítja a fejlécet. Az alapértelmezett *érték* = 1 beállításnál a fejléc minden egyes oszlopában az alapjául szolgáló mező neve jelenik meg.

A **HeadLines** tulajdonság a **DBGrid** vezérlőelembeni oszlopok felett egynél több soros szöveg megjelenítésének beállításához használható. Az alapértelmezett érték (1) az egyes oszlopokra a **Caption** tulajdonsággal közvetlenül módosítható.

Height

Szintaxis: *objektum*.**Height** [= *érték*]

A tulajdonság **Columns**, **Form**, **MDIForm**, **Picture**, **Printer** és **Screen** objektumon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox** és **VScrollBar** vezérlőelemen az adott objektum vagy vezérlőelem magasságának kiolvasására vagy beállítására használható. A tulajdonság a **Printer** és **Screen** objektumoknál a

programfejlesztési fázisban nem érhető el, futásidőben pedig csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az adott objektum vagy vezérlőelem magasságát megadó numerikus kifejezés.

A magasság mérése a következő feltételek szerint történik:

- **Form** objektumon — Az objektum külső (a keretet és címsort is magába foglaló) magassága twip mértékegységben. 1 (logikai) cm = 567 twip.
- **Picture** objektumon — A kép HiMetric egységben adott magassága. A HiMetric egység a **ScaleX** és **ScaleY** metódusokkal tetszés szerinti egységre konvertálható.
- **Printer** objektumon — A nyomtatóeszközre beállított valós, fizikai papírméret twip mértékegységben. A programfejlesztési fázisban nem érhető el. Futásidőben a tulajdonság értéke a **PaperSize** tulajdonság beállítása helyett használt.
- **Screen** objektumon — A képernyő magassága twip mértékegységben. A programfejlesztési fázisban nem érhető el, futásidőben csak olvasható.
- Vezérlőelemeken — A vezérlőelem magassága a vezérlőelem határának közepétől a különböző magasságokhoz megfelelően illeszkedő módon mérve. A tulajdonság a vezérlőelem befogadójának mértékegységét használja.

Az űrlapoknál és vezérlő objektumoknál a **Height** tulajdonság értéke követi azok méretének egy felhasználói művelet vagy programkód okozta változását. A tulajdonság egyes objektumokra érvényes maximális értéke rendszerfüggő.

A **Height** tulajdonságot a vele összefüggő **Left**, **Top** és **Width** tulajdonságokkal együtt egy objektum vagy vezérlőelem teljes területét érintő művelethez (mozgatásra, méretmódosításra vagy helyszükséglet számítására), a **ScaleLeft**, **ScaleTop**, **ScaleHeight** és **ScaleWidth** tulajdonságokkal együtt pedig a belső területét érintő számításhoz (másik objektum belsejében való rajzolásra vagy mozgatásra) használható.

Ha a **Height** tulajdonságot olyan nyomtatóvezérlőhöz állítjuk be, amely nem engedi meg ezen tulajdonság beállítását, a beállítás hibajelzés nélkül hatástalan lesz. A csak bizonyos értékeket elfogadó nyomtatóvezérlő esetén a beállítás hibajelzés nélkül az előírt érték lesz.

A **DriveListBox** és a **Style** = 0 (*Dropdown Combo*) vagy **Style** = 2 (*Dropdown List*) beállítású **ComboBox** esetén a **Height** tulajdonság nem módosítható.

HelpCommand

Szintaxis: *objektum*.**HelpCommand** [= *érték*]

A tulajdonság **CommonDialog** vezérlőelemen használható, és a közvetlen súgó hívási típusának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a

következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a súgó típusát meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	Értelmezése:
cdlCommand	&H102	Egy Help makrót hajt végre.
cdlContents	&H3	A Help témaköröket a .HPJ fájl [OPTION] részében a CONTENTS beállítással definiált módon jeleníti meg.
cdlContext	&H1	Egy adott témakapcsolat szerinti Help-információt jelenít meg. A témakapcsolatot a **HelpContext** tulajdonságban kell megadni.
cdlContextPopup	&H8	Egy előugró ablakban a .HPJ fájl [MAP] részében kapcsolatszámmal kijelölt Help-témát jeleníti meg.
cdlForceFile	&H9	A WinHelpben a megfelelő Help fájl megjelenítését biztosítja. Ha nem a megfelelő fájl van megjelenítve, a WinHelp a helyes fájlt nyitja meg.
cdlHelpOnHelp	&H4	A Help-információt a Help alkalmazás használatával jeleníti meg.
cdlIndex	&H3	A kívánt Help-fájl indexét jeleníti meg. Egy alkalmazás ezt az értéket csak egyetlen indexű Help-fájlokkal használhatja.
cdlKey	&H101	Egy adott kulcsszóra vonatkozó Help-információt jeleníti meg. A kulcsszót a **HelpKey** tulajdonságban kell megadni.
cdlQuit	&H2	A Help-alkalmazást értesíti, hogy az adott Help fájlra már nincs szükség.
cdlSetContents	&H5	Meghatározza az `F1` billentyű leütésekor megjelenítendő témát.
cdlPartialKey	&H105	Azt a kulcsszavak listájában talált témát jeleníti meg, amelynél a *dwData* paraméterben átadott kulcsszóhoz egy találat tartozik. Egynél több találat esetén a talált témák listáját a Go To mezőjében megadó Search párbeszédpanel jelenik meg. Ha nincs találat, csak a Search párbeszédpanel jelenik meg. A Search párbeszédpanelt kulcsszó megadása nélkül üres karakterlánc

		hosszú mutatójának használatával lehet megjeleníteni.
cdlSetIndex	&H5	A **HelpContext** tulajdonságban szereplő témakapcsolatot a **HelpFile** tulajdonságban megadott Help fájl aktuális indexévé teszi, amely mindaddig érvényes marad, amíg a felhasználó másik Help fájlt nem kíván elérni. Csak egynél több indexű Help fájlokhoz használható.

A **HelpCommand** tulajdonság konstansainak értékei az Object Browser CommonDialog Control (MSComDlg) objektumtárában találhatók.

HelpContext

Szintaxis: *objektum*.**HelpContext** [= *érték*]

A tulajdonság a **CommonDialog** vezérlőelemen, valamint az **Error** és az **Err** objektumon eltérő módon érvényes.

A **CommonDialog** vezérlőelemen a tulajdonság a kívánt Help-téma kapcsolatazonosítójának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a kapcsolatazonosítót meghatározó karakteres kifejezés.

A **HelpContext** tulajdonság a **cdlContext** beállítású **HelpCommand** tulajdonsággal együtt a megjeleníteni kívánt Help-téma meghatározására alkalmazható.

Az **Error** objektumon a tulajdonság egy Microsoft Help fájlbeli téma kapcsolatazonosítójának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a Help fájlbeli Help-téma érvényes azonosítója.

Ha egy Microsoft Windows Help fájl a **HelpFile** tulajdonságban adott, a **HelpContext** tulajdonság használatával az adott téma automatikusan megjeleníthető. Ha a **HelpFile** és a **HelpContext** tulajdonság egyaránt üres, a **Number** tulajdonság értéke kerül vizsgálatra, és ha az egy Visual Basic futásidejű hibaértéknek felel meg, a hibához tartozó kapcsolatazonosító lesz érvényes, egyébként pedig a Visual Basic Help fájl tartalma jelenik meg.

Az alkalmazásokban előforduló jellemző hibák kezelésére rutinok írhatók. Egy objektummal történő programozáskor a hibakezelés javítására vagy a felhasználónak egy

lekezelhetetlen hiba bekövetkeztéről történő tájékoztatására az objektumhoz tartozó Help fájl által adott súgó használható. OLE Automatizmus objektumok írásakor a súgót a programozók számára az objektummal történő programozáshoz, a felhasználót pedig a lekezelhetetlen hibára vonatkozó tájékoztatáshoz megfelelő alakra lehet formálni.

Az **Err** objektumon a tulajdonság egy Microsoft Help fájlbeli téma kapcsolatazonosítóját adja vissza. Használatára vonatkozóan az **Error** objektumra vonatkozóan leírtak értelemszerűen érvényesek.

HelpContextID

Szintaxis: *objektum*.**HelpContextID** [= *érték*]

A tulajdonság a **Form**, **MDIForm** és **MenuLine** objektumon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Menu**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemen érvényes, és egy objektumhoz tartozó társított kapcsolatszám kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá.
- A tulajdonság beállításához használható *érték* argumentum az *objektum* társított Help-témájának kapcsolatszámát meghatározó **Integer** numerikus kifejezés.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha az *érték*:	Értelmezése:
0	Nincs kapcsolatszám. (Alapértelmezett.)
> 0	Az érvényes kapcsolatszám.

Az alkalmazáson belül éppen használt objektumhoz kapcsolódó témaérzékeny súgóhoz ugyanazon kapcsolatszámot kell megadni az *objektum* és a Help fájl szerkesztésekor hozzá tartozó súgótéma számára.

Ha a Microsoft Windows operációs környezetben az alkalmazás számára már van létrehozott Help fájl és az alkalmazás **HelpFile** tulajdonsága beállított, az `F1` funkcióbillentyű leütésekor a Visual Basic automatikusan az aktuális kapcsolatszámnak, azaz a fókuszban lévő objektum **HelpContextID** értékének megfelelő témát jeleníti meg. Ha a **HelpContextID** értéke 0, a Visual Basic az objektum befogadójának **HelpContextID** tulajdonságát használja. Ha nem található nullától különböző kapcsolatszám, az `F1` billentyű leütése hatástalan lesz.

A **Menu** vezérlőelem esetén a **HelpContextID** tulajdonság rendes körülmények között írható és olvasható is, azonban az olyan menüknél csak olvasható, amelyeket a Visual Basic jelenít meg vagy ad át, mint például az Add-Ins menü Add-In Manager parancsa.

Az alkalmazásokhoz megfelelő Help fájlok a Visual Basic Professional Edition változathoz adott Microsoft Windows Help Compiler segítségével készíthetők.

HelpFile

Szintaxis: *objektum*.**HelpFile** [= *érték*]

A tulajdonság az **App**, **Err**, **Error**, **MenuLine** (Add-In) objektumon, valamint a **CommonDialog** vezérlőelemen eltérő módon érvényes.

Az **App** és **MenuLine** (Add-In) objektumon, valamint a **CommonDialog** vezérlőelemen a tulajdonság egy, az alkalmazás által a súgó vagy a közvetlen dokumentáció megjelenítésére használt Microsoft Windows Help fájl nevének és elérési útjának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az alkalmazáshoz tartozó Help fájl nevét és elérési útját meghatározó karakteres kifejezés.

A **HelpFile** tulajdonság a felhasználó által az Object Browser ablakból egy OLE kiszolgáló számára kért súgó esetén megjelenített Help fájl meghatározására is használható.

Az **Error** objektumon a tulajdonság egy Microsoft Help fájl teljes alakú elérési útját adja vissza. A tulajdonság használatára vonatkozó tudnivalók értelemszerűen azonosak az **Error** objektumon érvényes **HelpContext** tulajdonságnál leírtakkal.

Az **Err** objektumon a tulajdonság egy Microsoft Windows Help fájl teljes elérési útjának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a Help fájl teljes elérési útja.

A **HelpFile** tulajdonság használatakor általában érvényesek a következők:

Ha a Microsoft Windows operációs környezetben az alkalmazás számára már van létrehozott Help fájl és az alkalmazás **HelpFile** tulajdonsága beállított, az `F1` funkcióbillentyű leütésekor a Visual Basic automatikusan az aktuális kapcsolatszámnak, azaz a fókuszban lévő objektum **HelpContextID** értékének megfelelő témát jeleníti meg.

Az alkalmazásokhoz megfelelő Help fájlok a Visual Basic Professional Edition változathoz adott Microsoft Windows Help Compiler segítségével készíthetők.

HelpKey

Szintaxis: *objektum*.**HelpKey** [= *érték*]

A tulajdonság **CommonDialog** vezérlőelemen érvényes, és a kívánt Help-témát kijelölő kulcsszó kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a Help-témát kijelölő kulcsszót meghatározó karakteres kifejezés.

A **HelpKey** tulajdonságot a **cdlKey** beállítású **HelpCommand** tulajdonsággal együtt a megjelenítendő Help-téma kijelölésére lehet használni.

Hidden

Szintaxis: *objektum*.**Hidden** [= *érték*]

A **Hidden** tulajdonság **FileListBox** vezérlőelemen érvényes, és a vezérlőelemben listázandó fájlok **Hidden** attribútumát beállító **Boolean** érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A **FileListBox** vezérlőelemben a **Hidden** attribútumú fájlokat jeleníti meg. (Alapértelmezett)

False A **FileListBox** vezérlőelemben a nem **Hidden** attribútumú fájlokat jeleníti meg.

A tulajdonsággal a **FileListBox** vezérlőelemben megjelenítendő, az operációs környezetben használt standard fájlattribútum állítható be. Ha a tulajdonság beállítása futásidőben, programkódból történik, a **FileListBox** vezérlőelem újra beállítódik és azon csak az adott attribútumú fájlok jelennek meg.

HideSelection

Szintaxis: *objektum*.**HideSelection**

Ez a **TextBox** vezérlőelemen érvényes tulajdonság a kijelölt szöveg kiemelt megjelenését a vezérlőelem fókuszának elvesztésekor meghatározó értéket ad vissza. A szinta-

xisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság által visszaadott **Boolean** érték értelmezése a következő:

True A kijelölt szöveg a vezérlőelem fókuszának elvesztésekor nem kiemelten jelenik meg. (Alapértelmezett.)

False A kijelölt szöveg a vezérlőelem fókuszának elvesztésekor kiemelt megjelenésű marad.

A tulajdonság — például egy helyesírás-ellenőrző rutinban — arra használható, hogy egy szöveg kiválasztottságát akkor is jelezze, ha már más vezérlőelem került fókuszba.

HighLight

Szintaxis: *objektum*.**HighLight** [= *érték*]

Ez a **Grid** vezérlőelemen érvényes tulajdonság a kijelölt cellák kiemelt megjelenését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A kijelölt cella kiemelten jelenik meg. (Alapértelmezett)

False A kijelölt cella a kiválasztatlanokéval azonos, nem kiemelt megjelenésű marad.

Ha a **HighLight** tulajdonság értéke **False** és a felhasználó egy cellatartományt jelöl ki, a kijelölt tartomány vizuálisan változatlan marad.

hInstance

Szintaxis: *objektum*.**hInstance**

Ez az **App** objektumon érvényes tulajdonság az alkalmazás egy példányának kezelőjét adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **hInstance** tulajdonság értéke a Visual Basic 16-bites változatában egy **Integer**, a 32-bites változatban pedig egy **Long** típusú szám.

Egy projekttel a Visual Basic fejlesztési környezetben dolgozva a **hInstance** tulajdonság a Visual Basic példányának kezelőjét adja vissza.

HostName

Szintaxis: *objektum*.**HostName** [= *név*]

Ez az **OLE** befogadó vezérlőelemen érvényes tulajdonság a Visual Basic alkalmazás felhasználó által olvasható hostnevének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *név* argumentum a hostnevét megadó karakteres kifejezés.

Egy objektum szerkesztésekor a **HostName** tulajdonság értéke az objektumablak címében jelenhet meg, azonban az objektumot szolgáltató egyes alkalmazások nem jelenítik meg a **HostName** értékét.

hPal

Szintaxis: *objektum*.**hPal** [= *érték*]

A tulajdonság **Picture** objektumon érvényes tulajdonság az objektum képpalettájának kezelőjét megadó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a képhez tartozó paletta kezelője (HPAL).

A **hPal** tulajdonság a kezelő egy palettának DLL-ben lévő függvény vagy Windows API hívás részeként történő átadásakor használatos.

hWnd

Szintaxis: *objektum*.**hWnd**

Ez a tulajdonság **Form** és **MDIForm** objektumon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **ListBox**, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemen érvényes, és egy objektum vagy vezérlőelem kezelőjét adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A Microsoft Windows az alkalmazás minden egyes űrlapját és vezérlőelemét annak kezelőjével, azaz **hWnd** tulajdonságával azonosítja, és a Windows API hívásokkal együtt használja. Az aktív űrlap **hWnd** tulajdonsága számos Windows környezeti funkcióban argumentumként is használatos.

Hasonlóan a **hDC** tulajdonság használatához, ne felejtsük el, hogy a **hWnd** értéke a

program futása alatt is módosulhat, ezért az értékének változóban való tárolása helyett minden egyes felhasználáskor kell kiolvasni.

Felhívjuk a figyelmet arra, hogy a **hWnd** tulajdonság a Visual Basic 4.0-s verzióban már nem támogatja az **OLE** befogadó vezérlőelemet.

Icon

Szintaxis: *objektum*.**hWnd**

Ez a tulajdonság a **Form** és **MDIForm** objektumon érvényes, és futásidőben egy űrlap ikonállapotba hozásakor megjelenő, az űrlapot képviselő ikont ad vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az **Icon** tulajdonságot minden egyes futásidőben ikonállapotba hozható űrlaphoz a programfejlesztési fázisban lehet beállítani az ikon nevének a Properties mezőbe való beírásával. Az ikonnak .ICO kiterjesztésű és megfelelő formájú fájlban kell lennie. Ha nincs ikon meghatározva, az űrlap ikonállapotában a Visual Basic űrlapokra alapértelmezett ikonja jelenik meg.

Az űrlap funkcióját szemléltető, szabványos Windows formátumú ikonok a Visual Basic ICONS alkönyvtárában lévő Icon Library ikontárából választhatók. A futtatható fájl létrehozásakor az alkalmazáshoz rendelendő ikont az alkalmazás űrlapjának **Icon** tulajdonságán keresztül lehet beállítani.

Az űrlap ikonjának megjelenítéséhez az űrlapnak ikonállapotúnak, a **BorderStyle** tulajdonságnak 1 (*Fixed Single*) vagy 2 (*Sizable*), a **MinButton** tulajdonságnak pedig **True** értékűnek kell lennie.

Futásidőben egy objektum **Icon** tulajdonságának értéke egy másik tulajdonság **DragIcon** vagy **Icon** tulajdonságaként itt megadható. Az űrlap ikonja a **LoadPicture( )** függvény használatával is megadható: mivel a **LoadPicture( )** argumentum nélküli megadása egy üres ikont állít be az űrlapon, lehetőség van arra futásidőben történő rajzolásra is.

IgnoreNulls

Szintaxis: *objektum*.**IgnoreNulls** [= *érték*]

Ez az **Index** objektumon érvényes tulajdonság a rekordok indexbejegyzései között az indexmezőikben **Null** érték előfordulását jelző érték kiolvasására vagy beállítására használható. Objektumgyűjteményhez még nem tartozó új **Index** objektum esetén a tulajdonság írható és olvasható is, egyébként egy **Indexes** objektumgyűjtemény **Index** objektuma esetén csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az indexbejegyzése-

ket jelölő **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True **Null** értéket tartalmazó mezőknek nincs ndexbejegyzésük.
False **Null** értéket tartalmazó mezőknek van indexbejegyzésük.

Egy mező használatával történő gyors rekordkereséshez a mezőhöz indexet adhatunk meg. Ha egy indexelt mezőben **Null** értékű bejegyzéseket is megengedünk, és a várhatóan a bejegyzések legtöbbje **Null**, az **Index** objektum **IgnoreNulls** tulajdonságát **True** értékre állítva csökkenthető az indexek által használt tárolóterület.

Az **IgnoreNulls** és a **Required** tulajdonság beállításai a következők szerint határozzák meg, hogy a **Null** értékű rekordoknak van-e indexbejegyzésük:

IgnoreNulls	**Required**	**Null** az indexmezőben:
True	**False**	**Null** érték megengedett, nincs hozzáadott indexbejegyzés.
False	**False**	**Null** érték megengedett, az indexbejegyzés hozzáadott.
False vagy **True**	**True**	**Null** érték nem megengedett, nincs hozzáadott indexbejegyzés.

Image

Szintaxis: *objektum*.**Image**

Ez a tulajdonság **Form** objektumon és **PictureBox** vezérlőelemen érvényes, és a háttér állandó bitképéhez a Windows operációs környezet által szolgáltatott kezelőt adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Egy objektum **AutoRedraw** tulajdonsága meghatározza, hogy az objektum újrarajzolása a bitképes háttérgrafikával vagy a **Paint** eseményeken keresztül történik. A Windows operációs környezet egy objektum állandó bitképét az **Image** tulajdonsággal kiolvasható kezelő hozzá történő beállításával azonosítja.

Az **Image** tulajdonsága értéke az **AutoRedraw** beállítástól függetlenül létezik. Ha az **AutoRedraw** tulajdonság értéke **True** és még nem készült rajz, a kép csak a **BackColor** tulajdonsággal beállított színt és a grafikát jeleníti meg.

Az **Image** tulajdonság értékét a **Picture** tulajdonsághoz lehet beállítani. Az **Image** a Windows API hívások számára is átad egy értéket.

Az **Image**, **DragIcon** és **Picture** tulajdonságokat közvetlenül kell használni egyéb tulajdonságok értékének beállításához, a **SavePicture** utasítással történő mentéshez vagy valamely elemnek a vágólapra történő átvitelére. Ezeket a tulajdonságokat nem lehet ideiglenes változóként beállítani, majd az eredményeket azután használni.

Az **AutoRedraw** tulajdonság az **Image** tulajdonság, vagyis a bitkép kezelőjének megváltoztatását okozhatja. Amikor az **AutoRedraw** értéke **True**, az objektum **hDC**

tulajdonsága az **Image** által visszaadott bitképet tartalmazó eszközkapcsolat kezelőjévé válik.

Index

Szintaxis: *objektum* [(*érték*)].**Index**

Ez a tulajdonság objektumokon és — eltérő értelmezéssel — vezérlőelemeken is érvényes.

A tábla típusú **Recordset** (és az elavult **Table**) objektumon az **Index** tulajdonság az objektumban az aktuális **Index** objektum nevét meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a tábla típusú **Recordset** (és az elavult **Table**) objektum **TableDef** objektumának **Indexes** halmazában egy **Index** objektum nevét meghatározó **String** adattípusú kifejezés.

A rekordok az alaptáblában a beírás sorrendjében léteznek, az onnan indexelés nélkül történő olvasáskor sorrendjük nem előírt lesz. Az **Index** tulajdonság beállításával az adatbázis által visszaadott rekordok sorrendjét anélkül lehet módosítani, hogy az hatással lenne a tárolt rekordok sorrendjére, mivel az csak a **Table** objektummal történő elérési módjukat határozza meg egy adott index kiválasztásakor vagy a **Recordset** objektumok Microsoft Jet adatbázis-szerkesztő általi létrehozásakor.

Az **Index** objektumban minden egyes **Field** objektum **Attributes** tulajdonsága meghatározza a rekordok sorrendjét, és ebből következően az index elérési módját.

A megadott **Index** objektumnak már létezőnek kell lennie. Nem létező **Index** objektumhoz beállított **Index** tulajdonság vagy a **Seek** metódus használatakor nem beállított **Index** tulajdonság kezelhető hibát okoz.

Egy **TableDefs** objektum **Indexes** halmazának vizsgálatával határozható meg, hogy melyik **Index** objektum érhető el az adott **TableDefs** objektumból létrehozott tábla típusú **Recordset** számára.

A tábla számára új indexet egy új **Index** objektum létrehozásával, tulajdonságainak beállításával, az alapjául szolgáló **TableDef** objektum **Indexes** halmazához csatolásával, majd a **Recordset** újbóli megnyitásával lehet létrehozni.

A **Table** objektumból visszaadott rekordok csak az alapjául szolgáló **TableDef** számára érvényes index szerint rendezhetők. Ha a rekordokat más szabály szerint kívánjuk rendezni, egy **ORDER BY** záradékot tartalmazó SQL utasítást használva kell a **Sort** tulajdonsággal adott rendezési módot használó eredményhalmaz vagy pillanatfelvétel típusú **Recordset** objektumot megnyitni.

A táblákhoz nem kötelező indexeket létrehozni, azonban másként a nagyméretű,

nem indexelt tábla adott rekordjához történő hozzáférés vagy egy **Recordset** létrehozása sok időt igényel. Ugyanakkor túl sok index létrehozása lelassítja a frissítési, csatolási és törlési műveleteket, mivel mindegyik index automatikusan frissítődik.

Vezérlőelemtömb **CheckBox**, **ComboBox**, **CommandButton**, **CommonDialog**, **Data**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **Line**, **ListBox**, **Menu**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox**, **Timer** és **VScrollBar** vezérlőelemein az **Index** tulajdonság a vezérlőelemet a tömbben azonosító érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az egyedi vezérlőelemet egy vezérlőelemtömbben azonosító értéket meghatározó **Integer** típusú numerikus kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha az *érték*:	Értelmezése:
Nincs	A vezérlőelem nem eleme egy tömbnek. (Alapértelmezett.).
0 — 32767	A vezérlőelem egy tömb eleme és az érték a vezérlőelemnek a tömbön belüli pozícióját adja meg. A tömbön belül minden egyes vezérlőelem azonos **Name** tulajdonságú. A tömb következő érvényes indexét a Visual Basic automatikusan állítja be.

Ha egy űrlapon több vezérlőelemi objektum **Name** tulajdonsága azonos, azok automatikusan egyazon vezérlőelemtömb elemei lesznek, és a programkódban az egyes elemekre a nevük után zárójelbe tett **Index** tulajdonságukkal kell hivatkozni, például a `Gomb(4)` alakban. A tömb egyes elemeinek megkülönböztetésére a **Tag** tulajdonságuk értékét is felhasználhatjuk.

Ha egy tömbbeli vezérlőelem egy eseményt érzékel, a Visual Basic a tömb eseményeljárását a megfelelő **Index** tulajdonság értékét járulékos argumentumként használva hívja meg. Az **Index** tulajdonság másik alkalmazási lehetősége vezérlőelemnek futásidőben, a **Load** utasítással történő dinamikus létrehozása vagy az **Unload** utasítással történő eltávolítása.

Noha a Visual Basic automatikusan beállítja az egymás után következő vezérlőelemek **Index** tulajdonságának értékét, ez felülírható és akár egyes egész értékek át is ugorhatók, így például a tömb első elemének **Index** értéke is tetszőleges, nem 0 érték lehet. A programkódban az **Index** értékkel a tömbbe nem tartozó vezérlőelemre történő hivatkozás futásidejű hibát okoz.

Tömbből vezérlőelemet a **Name** tulajdonságának megváltoztatásával lehet kiemelni, egyedi vezérlőelemmé tenni.

Inherit

Szintaxis: *objektum*.**Inherit**

Ez a tulajdonság **Container** objektumokon érvényes, és az alapértelmezett **Permission** tulajdonság beállításának az új **Document** objektum általi öröklését meghatározó **Boolean** típusú értéket ad vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság által visszaadott értékek és értelmezésük a következő:

True	Az új objektum örökli a jogosultságokat.
False	Az új objektum nem örökli a jogosultságokat.

Az **Inherit** tulajdonságot a **Permission** tulajdonsággal összhangban annak meghatározására lehet használni, hogy az új dokumentumok létrehozásukkor automatikusan milyen jogosultságokkal rendelkezzenek. Ha az **Inherit** tulajdonságnak **True** értéket adunk, majd beállítjuk a jogosultságot egy befogadó számára, a befogadóban létrehozott minden új dokumentum ezt a jogosultságot fogja örökölni.

Az **Inherit** tulajdonság beállítása nem hat a **Container** már létező dokumentumaira, vagyis az **Inherit** tulajdonsággal nem lehet azok jogosultságait módosítani.

Inherited

Szintaxis: *objektum*.**Inherited**

Ez a tulajdonság **Container** és **Property** objektumokon érvényes, és a **Property** objektumnak az alapjául szolgáló objektumból való öröklődését meghatározó érték kiolvasására használható. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság által visszaadott értékek és értelmezésük a következő:

True	A **Property** objektum öröklött.
False	A **Property** objektum nem öröklött. Az előre definiált tulajdonságokat jelölő beépített **Property** objektumok esetén a tulajdonság csak **False** lehet.

Az **Inherited** tulajdonságot annak meghatározására lehet használni, hogy egy felhasználó által definiált tulajdonság az azt használó objektum vagy egy másik objektum számára volt-e létrehozva.

IniPath

Szintaxis: *objektum*.**IniPath** [= *érték*]

Ez a tulajdonság **DBEngine** objektumon érvényes, és a Microsoft Jet adatbázisszerkesztő inicializációs (.INI) fájl nevét és teljes alakú elérési útjának kiolvasására vagy

beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a telepíthető ISAM adatbázishoz szükséges Jet beállításokat vagy paramétereket tartalmazó, felhasználó által megadott ASCII fájl elérési útját tartalmazó **String** típusú kifejezés.

A Jet egy felhasználó által megadott inicializálási fájlnak megfelelően konfigurálható, így a fájl használatával a Jet működését meghatározó beállítások adhatók meg, vagy a telepíthető ISAM (például a Paradox) adatbázisok számára lehet inicializálási paramétereket átadni.

A Jet inicializálása a 16-bites és 32-bites rendszerekben eltérő:

- A 16-bites rendszerben az inicializálás akkor hatásos, ha az **IniPath** tulajdonság beállítása bármely adathozzáférési funkciónak a rendszer valamely alkalmazása által történő meghívása előtt megtörtént. Ha a Jet ekkor már inicializált lenne, kezelhető hiba következik be. A beállítás hatóköre Public, és ha már beállított, a Jetet használó minden program újraindítása nélkül nem módosítható.
- A 32-bites rendszerben az inicializálás akkor hatásos, ha az **IniPath** tulajdonság beállítása bármely adathozzáférési funkciónak az alkalmazásunk által történő meghívása előtt megtörtént. Ha a Jet ekkor már inicializált lenne, kezelhető hiba következik be. A beállítás hatóköre csak az alkalmazásunkra korlátozódik, és ha már beállított, az alkalmazásunk újraindítása nélkül nem módosítható.

A Microsoft Access 1.1-s verziója és a Visual Basic 4.0-s verziója nem azonos Jet verziót használ, azonban más, a Microsoft Jet 2.5/3.0 verzióját használó alkalmazások inicializálják az adatbázis-szerkesztőt.

Az **IniPath** tulajdonság helyettesíti a Visual Basic 3.0-s verzióban használt — itt már elavult — **SetDataAccessOption** utasítás DB_OPTIONINIPATH paraméterét.

InitDir

Szintaxis: *objektum*.**InitDir** [= *elérési_út*]

A tulajdonság a **CommonDialog** (File Dialog) vezérlőelemen érvényes, és a kiindulási fájltár kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *elérési_út* argumentum a kiindulási fájltárat meghatározó karakteres kifejezés.

Ha a tulajdonságot nem állítjuk be, az éppen aktuális könyvtár lesz érvényes.

Instancing

Szintaxis:

A futásidőben nem elérhető tulajdonság a **ClassModule** objektumon érvényes, és egy **Public** hatókörű osztály projekten kívüli példányának létrehozási lehetőségét meghatározó érték beállítására használható.

A tulajdonság lehetséges értékei a következők:

Ha az **Instancing**:	Értelmezése:
0	*Not Creatable* — Az osztály példányát csak a projekten belül lehet létrehozni. (Alapértelmezett.)
1	*Creatable SingleUse* — Az osztály példányát a projekten belül és azon kívül is létre lehet hozni. Az osztály egy példányának projekten kívüli OLE ügyfél általi minden egyes kérése esetén az OLE kiszolgáló egy különálló példánya indul el.
2	*Creatable MultiUse* — Az osztály példányát a projekten belül és azon kívül is létre lehet hozni. Az osztály egy példányának projekten kívüli OLE ügyfél általi minden egyes kérése esetén az OLE kiszolgáló egy már futó példánya (ha van ilyen) lesz használt. Ha nincs futó OLE kiszolgáló, az osztály szolgáltatásához egy másolata indul el.

Egy létrehozható osztály egy példányát a **CreateClass( )** függvénnyel vagy a **Dim** utasítással tudjuk létrehozni egy másik alkalmazásból.

Ha a **Public** tulajdonság értéke **False**, az **Instancing** tulajdonság beállítása hatástalan és az osztály példányai mindig létrehozhatók az osztályt megadó projekten belül. Ha a **Public** tulajdonság értéke **True**, az osztály látható és amint egy példánya létezik, más alkalmazások által vezérelhetővé válik.

IntegralHeight

Szintaxis: *objektum*.**IntegralHeight** [= *érték*]

Ez a futásidőben csak olvasható tulajdonság **DBComboBox** és **DBList** vezérlőelemen érvényes, és a vezérlőelem részleges elemek megjelentetési működését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a lista átméretezhetőségét meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A lista csak a teljes elemek megjelenítéséhez megfelelően méretezi át magát. (Alapértelmezett.)

False A lista még akkor sem méretezi át magát, ha az elem túl hosszú a teljes megjelenítéshez.

Interval

Szintaxis: *objektum*.**Interval** [= *időtartam*]

Ezt a **Timer** vezérlőelemen érvényes tulajdonságot a vezérlőelem két **Timer** eseményének meghívása közötti időtartam kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *időtartam* argumentum az időközt milliszekundumban megadó numerikus kifejezés.

Az *időtartam* argumentum lehetséges értékei és értelmezésük a következő:

Ha az időtartam:	Értelmezése:
0	A **Timer** vezérlőelem letiltott. (Alapértelmezett.)
1 — 65535	A **Timer** vezérlőelem **Enabled** tulajdonságának **True** értéke esetén érvényes időköz milliszekundumban. A beállítható legnagyobb időköz kb. 1 perc (65,535 másodperc).

Az **Interval** tulajdonság a programfejlesztési fázisban és futásidőben egyaránt beállítható, és használata során a következőkre kell figyelemmel lenni:

- Az időtartam beírása után a **Timer** vezérlőelem indítása annak **Enabled** tulajdonságának **True**, leállítása pedig **False** értékre állításával lehetséges. A **Timer** engedélyezését követően az időtartam visszaszámolása az **Interval** tulajdonságban adott értéktől indul.
- Az **Interval** tulajdonság után az időtartam lejárta után végrehajtandó műveleteket tartalmazó **Timer** eseményeljárást lehet elhelyezni.
- A Windowsban legfeljebb 16 időzítőt használhatunk. Nem létező időzítő indítása futásidejű hibát okoz.

IsolateODBCTrans

Szintaxis: *objektum*.**IsolateODBCTrans** [= *érték*]

Ez a tulajdonság **Workspace** objektumon érvényes, és az ugyanazon ODBC adatbázist érintő többszörös tranzakciók elszigeteltségét meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a tranzakciók elszigeteltségét meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True Az azonos ODBC adatbázist érintő tranzakciók elszigeteltek.
False Az azonos ODBC adatbázist érintő tranzakciók nem elszigeteltek. (Alapértelmezett.)

Bizonyos helyzetben ugyanazon ODBC adatbázison egyidőben többszörös tranzakciókra van szükség. Ehhez mindegyik tranzakció számára egy különálló **Workspace** objektumot kell megnyitni. Noha mindegyik **Workspace** objektum saját ODBC kapcsolattal rendelkezik, ez jelentősen lelassítja a rendszer működését. Mivel a tranzakciók elkülönítése rendes körülmények között szükségtelen, az ugyanazon felhasználó által többszörös **Workspace** objektumokból megnyitott ODBC kapcsolatok alapértelmezetten megosztottak.

Néhány ODBC kiszolgáló — például a Microsoft SQL Server — nem engedi meg az egyidejű tranzakciókat egy kapcsolaton belül, ehhez minden egyes **Workspace** objektum megnyitásakor az **IsolateODBCTrans** tulajdonságot **True** értékre kell beállítani, így mindegyik **Workspace** objektumhoz különálló ODBC kapcsolatot létrehozva.

Italic

Szintaxis: *objektum*.**Italic** [= *érték*]

Ez a **Font** objektumon érvényes tulajdonság az objektum karaktertípusának dőlt megjelenését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a dőlt megjelenést meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A karakterek dőlt megjelenésűek lesznek.
False A karakterek nem dőlt megjelenésűek lesznek. (Alapértelmezett.)

Mivel a **Font** objektum a programfejlesztési fázisban közvetlenül nem érhető el, **Italic** tulajdonságának értékét a kívánt vezérlőelem Properties ablakban a Font tulajdonság kiválasztásával és a Properties gombra kattintással, majd a Font párbeszédpanel Font Style mezőjében az Italic vagy Bold Italic kiválasztásával állíthatjuk be. A tulajdonság futásidőben történő beállítása közvetlenül, a **Font** objektum számára a szükséges értéket megadva történhet.

ItemData

Szintaxis: *objektum*.**Index**(*index*) [= *érték*]

Ez a tulajdonság **ComboBox** és **ListBox** vezérlőelemeken érvényes, és a vezérlőelembeli minden egyes elemhez tartozó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum az objektum meghatározott elemének száma.
- A tulajdonság beállításához használható *érték* argumentum a meghatározott elemhez kapcsolandó szám.

Az **ItemData** tulajdonság **Long** típusú értékekből álló és a vezérlőelem **List** tulajdonságának értékével azonos számú elemből álló tömb, a tömb elemeivel pedig a programon belül a vezérlőelem hozzájuk rendelt listaelemeire lehet hivatkozni, azaz a tulajdonság egy lista és egy adattömb elemeit egymáshoz rendelő indexként használható.

Ha a listához az **AddItem** metódussal egy új elemet adunk, az elem automatikusan az **ItemData** tömbben is megjelenik. Mivel az **ItemData** új eleme nem nulla inicializálási értéket kap, hanem megtartja az ahhoz a pozícióhoz a hozzáadás előtt tartozott értéket, ezért az **AddItem** művelet után mindig egy **ItemData** értékbeállítást is kell végezni.

KeepLocal

Szintaxis: *objektum*.**KeepLocal** [= *érték*]

Ez a **Database** objektumon érvényes tulajdonság az objektum egyes értékeinek a másolatkészítéskor lokálisként való megtartását meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság csak az adatbázisnak sokszorosítható űrlappá konvertálása előtt állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a lokális értékek megőrzését meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True	Az objektum a lokális értékeket megtartja a másolás során.
False	Az objektum nem tartja meg a lokális értékeket a másolás során. (Alapértelmezett.)

Amikor egy nem sokszorosítható adatbázis másolható űrlappá konvertálódik, — hacsak külön nem jelezzük — az adatbázis minden egyes objektuma is sokszorosíthatóvá válik. A konverzió előtt a nem sokszorosíthatónak kívánt (például bizalmas adatokat

tartalmazó) objektumon létrehozható és beállítható a **KeepLocal** tulajdonság, így ez nem másolódik. A tulajdonság nem használható a már másolható űrlappá konvertálódott objektumokon.

Miután egy adatbázis sokszorosíthatóvá vált, minden új, az alappéldányban vagy a halmaz más másolatánál létrehozott objektum lokális objektumként kezelt lesz, vagyis ezek abban a másolatban maradnak, amelyben létrehozásuk történt és nem kerülnek terjesztésre a halmazon belül. A készletben minden egyes új másolat létrehozásakor a forráspéldány összes sokszorosítható objektuma átkerül a másolatba, azonban a lokális objektumok nem.

Lehetnek olyan példányok egy másolatban, amelyben a létrehozott új objektumot minden felhasználó számára használhatóvá kívánunk tenni. A lokális objektum másolhatóvá tételéhez azt az alappéldányban kell létrehozni vagy abba kell importálni, majd ott a **Replicable** tulajdonságát **True** értékre kell beállítani.

Az az objektum, amelynek **KeepLocal** vagy **Replicable** tulajdonságát be kívánjuk állítani, ezeket a tulajdonságokat más objektumtól örökölhette, azonban a tulajdonságok értékeit nem, azt minden egyes objektumra közvetlenül kell beállítani.

KeyPreview

Szintaxis: *objektum*.**KeyPreview** [= *érték*]

Ez a **Form** objektumon érvényes tulajdonság a billentyűzetesemények (**KeyDown**, **KeyPress** és **KeyUp**) űrlapra vagy a vezérlőelemekre történő meghívásának rendjét meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az események fogadási módját meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True	Elsőként az űrlap, majd az aktív vezérlőelem kapja meg a billentyűzetesemény t.
False	Csak az aktív vezérlőelem kapja meg a billentyűzeteseményt, az űrlap nem. (Alapértelmezett.)

A **KeyPreview** tulajdonság általában egy űrlap billentyűzetkezelő eljárásában használható, például a billentyűzeteseményeket fogadó vezérlőelemek mindegyikéhez írt egyedi programok helyett a funkcióbillentyűk használatának űrlapszintű feldolgozására.

Ha az űrlapon nincs látható és engedélyezett vezérlőelem, a billentyűzeteseményt automatikusan az űrlap fogadja.

A billentyűzetesemény űrlapszintű kezeléséhez és a vezérlőelem billentyűzetesemény-fogadási viselkedésének letiltásához az űrlap **KeyPress** eseményeljárásában

ANSI_kód argumentumként, a **KeyDown** eseményeljárásában pedig *billentyűkód* argumentumként egyaránt 0 értéket kell megadni.

Megemlítésre érdemes, hogy egyes vezérlőelemek képesek a billentyűzeteseményeket úgy elfogni, hogy azokat az űrlap nem kapja meg.

LargeChange

Szintaxis: *objektum*.**LargeChange** [= *érték*]

Ez a tulajdonság a **HScrollBar** és **VScrollBar** vezérlőelemeken érvényes, és a vezérlőelem **Value** tulajdonságának a felhasználó által a gördítőcsúszka és a gördítőnyilak közötti területre kattintáskor bekövetkező értékváltozás nagyságának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **Value** tulajdonság értékváltozását meghatározó, 1 és 32767 közötti (alapértelmezetten 1) **Integer** szám.

A Windows operációs környezet a **Form** objektumon, valamint a **ComboBox** és **ListBox** vezérlőelemeken az adatmennyiségtől függően automatikusan állítja be az arányos gördítési lépésközt, azonban a **HScrollBar** és **VScrollBar** vezérlőelemeken ezt a felhasználási követelményekkel összhangban magunknak kell elvégeznünk.

A **HScrollBar** és **VScrollBar** vezérlőelemek maximális és minimális tartományai a **Max** és **Min** tulajdonságokkal állíthatók be.

A tulajdonság beállítása jellemzően a programfejlesztési fázisban történik, azonban lehetséges a futásidőben történő dinamikus beállítás is.

LastDLLError

Szintaxis: *objektum*.**LastDLLError**

Ez a tulajdonság csak a 32-bites Windows operációs környezetben használható, és egy DLL-beli hívás esetén bekövetkező rendszerhiba számát adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság csak a Visual Basic kódban **Declare** utasítást kívánó DLL hívásokra alkalmazható. Egy ilyen híváskor a hívott függvény rendszerint egy, a végrehajtás sikerességét vagy hibáját jelző értéket ad vissza és a **LastDLLError** tulajdonság feltöltött lesz. A visszaadott érték a DLL függvény dokumentációja alapján értelmezhető. Amikor hibakód visszaadása történik, a Visual Basic azonnal megvizsgálja a **LastDLLError** tulajdonságot. Ha az beállított, nem merül fel kivétel.

LastModified

Szintaxis: *objektum*.**LastModified**

A tulajdonság eredményhalmaz és tábla típusú **Recordset** (valamint az elavult **Dynaset**, **Snapshot** és **Table**) objektumon használható, és egy, a legutóbb csatolt vagy módosított rekord könyvjelzőjét adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság visszaadott értéke **Variant** vagy **String** típusú lehet és a legutolsóként csatolt vagy módosított rekordra mozgáshoz használható.

LastUpdated

Szintaxis: *objektum*.**LastUpdated**

A tulajdonság **Document**, **QueryDef**, tábla típusú **Recordset**, **TableDef** (valamint az elavult **Table**) objektumon érvényes, és az objektumon alapuló alaptábla legutolsó módosításának dátumát és időpontját a **Variant** adattípus **VarType** 7 (Date) altípusa szerinti értékként adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A dátum és az időpont értékek abból a számítógépből származnak, amelyen az alaptábla utolsó frissítése történt. A többfelhasználós környezetben a felhasználók a **LastUpdated** tulajdonság értékét közvetlenül a fájlkiszolgálótól kaphatják meg.

LBound

Szintaxis: *objektum*.**LBound**

A tulajdonság vezérlőelemek tömbjén érvényes, és a tömbben használt legkisebb sorszámot adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az **LBound** tulajdonság a tömbben legelső elem **Index** tulajdonságának értékével azonos. Ez az érték jellemzően 0, mivel a Visual Basic a tömb első elemének automatikusan ezt az értéket adja.

Left

Szintaxis: *objektum*.**Left** [= *érték*]

Ez a tulajdonság **Form**, **Column** és **MDIForm** objektumon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **CommonDialog**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox**, **Timer** és **VScrollBar** vezérlőelemeken érvényes, és az adott objektum belső bal oldali éle és a be-

fogadójának bal oldali éle közötti távolság kiolvasására vagy beállítására használható. A tulajdonság **Timer** vezérlőelemen futásidőben nem elérhető. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a távolságot megadó **Single** típusú numerikus kifejezés.

A **Left** tulajdonság értéke űrlapon twip egységben, a vezérlőelemeken pedig a befogadó koordinátarendszere által meghatározott egységben értelmezett. 1 (logikai) cm = 567 twip. A tulajdonság értéke követi az objektumnak egy felhasználói művelet vagy programkód okozta helyváltozását.

A **Left** tulajdonságot a vele összefüggő **Height**, **Top** és **Width** tulajdonságokkal együtt egy objektum vagy vezérlőelem teljes területét érintő művelethez (mozgatásra, méretmódosításra vagy helyszükséglet számítására), a **ScaleLeft**, **ScaleTop**, **ScaleHeight** és **ScaleWidth** tulajdonságokkal együtt pedig a belső területét érintő számításhoz (másik objektum belsejében való rajzolásra vagy mozgatásra) használható. A mértékegységfüggő tulajdonságok csak a **Form** és **Printer** objektumokon, valamint a **PictureBox** vezérlőelemeken használhatók.

LeftCol

Szintaxis: *objektum*.**LeftCol** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **DBGrid** és **Grid** vezérlőelemeken érvényes, és egy vezérlőelemben a bal oldali legszélső látható oszlop kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a bal oldali legszélső látható oszlop számát megadó numerikus kifejezés.

A **LeftCol** tulajdonság programkódban történő használatával a **DBGrid** és **Grid** vezérlőelemek programozottan görgethetők. Ugyanezen vezérlőelemek legfelső látható sorát a **TopRow** tulajdonsággal lehet meghatározni.

A beállítható legnagyobb szám a **Grid** vezérlőelemben látható oszlopok száma -1 érték lehet, az ennél nagyobb szám megadása hibát okoz.

LegalCopyright

Szintaxis: *objektum*.**LegalCopyright**

A futásidőben csak olvasható tulajdonság **App** objektumon használható, és a futó

alkalmazás szerzői jogi információit tartalmazó karakterláncot adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság karakterláncát a programfejlesztési fázisban az EXE Options párbeszédpanelen lehet beállítani.

LegalTrademarks

Szintaxis: *objektum*.**LegalTrademarks**

A futásidőben csak olvasható tulajdonság **App** objektumon használható, és a futó alkalmazás védjegyére vonatkozó jogi információit tartalmazó karakterláncot adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság karakterláncát a programfejlesztési fázisban az EXE Options párbeszédpanelen lehet beállítani.

LinkItem

Szintaxis: *objektum*.**LinkItem** [= *érték*]

Ez a tulajdonság **Label**, **PictureBox** és **TextBox** vezérlőelemeken érvényes, és egy másik alkalmazással folyó DDE párbeszédben a fogadó vezérlőelem számára átadandó adat kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a fogadó vezérlőelem számára átadandó adatot tartalmazó karakteres kifejezés.

A tulajdonság a standard alakú, az *alkalmazás*, *téma* és *elem* argumentumokat használó DDE szintaxisban használt *elem* argumentumnak felel meg. Beállításához az alkalmazás által felismerhető adategységet kell hivatkozásként megjelölni. A teljes adatkapcsolatot a **LinkItem** és **LinkTopic** együttes használatával lehet megadni a forrásalkalmazástól a fogadó vezérlőelemhez, majd a kapcsolatot a **LinkMode** tulajdonsággal beállításával lehet aktiválni.

A **LinkItem** csak egy fogadóként használt vezérlőelemre állítható be. Amikor egy Visual Basic űrlap egy DDE kapcsolatban forrásként szerepel, az azon lévő bármely **Label**, **PictureBox** vagy **TextBox** vezérlőelem neve a fogadó által használt *alkalmazás*|*téma*!*elem* karakterláncbeli *elem* argumentum lehet.

Mivel a DDE vezérlőelem egyidejűleg szerepelhet forrásként és fogadóként is, könnyen végtelen hurok alakulhat ki. Ennek elkerülésére az alkalmazásokban a fogadóforrás és forrás-fogadó kapcsolat mindkét irányában rokon, de nem azonos elemeket kell használni.

A programfejlesztési fázisban az Edit menü Paste Link parancsával kialakított állandó adatkapcsolat a **LinkMode**, a **LinkTopic** és **LinkItem** tulajdonságot is beállítja. Ezt követően a kapcsolat az űrlapokkal együtt tárolódik, és az űrlap minden egyes betöltése után a Visual Basic megkísérli annak újbóli létrehozását.

LinkMode

Szintaxis: *objektum*.**LinkMode** [= *érték*]

Ez a tulajdonság **Form** és **MDIForm** objektumon, valamint **Label**, **PictureBox** és **TextBox** vezérlőelemeken érvényes, és egy másik alkalmazással folyó DDE párbeszédben használt kapcsolattípus kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a kapcsolat típusát meghatározó Integer érték.

A tulajdonság a DDE kapcsolatot a következők szerint aktiválja:

- Vezérlőelemen — A Visual Basic űrlapon lévő fogadó objektum a vezérlőelem **LinkTopic** és **LinkItem** tulajdonságai által beállított módon kezdeményezhet adatcserét.
- Űrlapon — A fogadó alkalmazás a Visual Basic forrás űrlapjával az *alkalmazás|téma!elem* kifejezésében foglaltaknak megfelelően kezdeményezhet adatcserét.

A DDE kapcsolatban fogadóként használt vezérlőelemeknél az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha az *érték*:	Értelmezése:
0	*None* — Nincs DDE kapcsolat. (Alapértelmezett.)
1	*Automatic* — A fogadó vezérlőelem a csatolt adat minden változásakor frissítődik.
2	*Manual* — A fogadó vezérlőelem csak a **LinkRequest** metódus meghívásával frissíthető.
3	*Notify* — A csatolt adat minden megváltozása egy **LinkNotify** eseményt eredményez, azonban a fogadó vezérlőelem csak a **LinkRequest** metódus meghívásával frissíthető.

A DDE kapcsolatban forrásként használt űrlapoknál az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha az *érték*:	Értelmezése:
0	*None* — Nincs DDE kapcsolat, nincs a kiszolgálóval DDE kapcsolatot kezdeményező fogadó alkalmazás vagy az űrlapba adatot helyezni kívánó alkalmazás. A programfejlesztési fázisban beállított **LinkMode** = 0 érték a program futása alatt nem módosítható. (Alapértelmezett.)
1	*Source* — Egy űrlapon lévő minden **Label**, **PictureBox** és **TextBox** vezérlőelemeken lehetővé teszi, hogy adatot adjon át bármelyik, DDE kapcsolaton át az űrlappal adatcserét kezdeményező fogadó számára. Ha ilyen kapcsolat létezik, a Visual Basic automatikusan értesíti a fogadót a vezérlőelem tartalmában bekövetkező változásáról. A fogadó alkalmazás is képes adatot helyezni az űrlapon lévő bármelyik **Label**, **PictureBox** vagy **TextBox** vezérlőelembe. A programfejlesztési fázisban beállított **LinkMode** tulajdonság értéke a program futása alatt módosítható.

A **LinkMode** tulajdonsághoz még további alkalmazási feltételek is kapcsolódnak:

- Ha a fogadó vezérlőelemekre vonatkozó **LinkMode** tulajdonsága nem 0, a Visual Basic a **LinkTopic** és **LinkItem** tulajdonságban meghatározott kapcsolatot kezdeményez, a kiszolgáló pedig a kapcsolat típusától (*Automatic*, *Manual*, *Notify*) függően frissíti a fogadó vezérlőelemét.
- Ha egy **Label**, **PictureBox** és **TextBox** vezérlőelemek által egy forrás alkalmazással kezdeményezett automatikus (**LinkMode** = 1) kapcsolat, minden egyes új adat érkezése a vezérlőelemen egy **Change** eseményt is előidéz, függetlenül attól, hogy az új adat eltér vagy sem a vezérlőelem jelenlegi tartalmától. Ez az egyetlen mód, hogy **Change** esemény következhessen be egy **Label** vezérlőelemen.
- Ha a forrás alkalmazás bezár egy Visual Basic fogadó vezérlőelemmel folyó adatcserét, az adott vezérlőelem **LinkMode** tulajdonság értéke 0 (*None*) lesz.
- Ha egy űrlap **LinkMode** tulajdonságának értékét a programfejlesztési fázisban alapértelmezettként hagyjuk, a 0 (*None*) érték a programfutás során már nem módosítható. Ha az űrlapot forrásként akarunk használni, a **LinkMode** tulajdonságának értékét már a fejlesztési fázisban ennek megfelelően (**LinkMode** = 1) kell beállítani, majd a programfutás alatt a mód szükség szerint megváltoztatható.

A programfejlesztési fázisban az Edit menü Paste Link parancsával kialakított állandó adatkapcsolat a **LinkMode**, a **LinkTopic** és **LinkItem** tulajdonságot is beállítja. Ezt követően a kapcsolat az űrlapokkal együtt tárolódik, és az űrlap minden egyes betöltése után a Visual Basic megkísérli annak újbóli létrehozását.

LinkTimeout

Szintaxis: *objektum*.**LinkTimeout** [= *érték*]

Ez a tulajdonság **Label**, **PictureBox** és **TextBox** vezérlőelemeken érvényes, és egy vezérlőelem DDE üzenetre várakozásának időtartamát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a várakozási időt tizedmásodpercben meghatározó numerikus kifejezés. Alapértelmezett értéke 50 (azaz 5 másodperc), a beállítható maximális időtartam pedig 65535, azaz kb. 109 perc lehet.

A DDE kapcsolaton belüli várakozási idő az egyes alkalmazásoktól függően eltérő lehet. A megfelelő idő beállításával elkerülhető, hogy a Visual Basic feleslegesen adjon hibajelzést a túl hosszú várakozási idő miatt.

Ha a **LinkTimeout** tulajdonság értéke -1, a DDE kapcsolatban a vezérlőelem válaszra várakozási ideje a maximális érték (65535) lesz. A felhasználó a várakozást az ESC billentyű leütésével szakíthatja meg.

LinkTopic

Szintaxis: *objektum*.**LinkTopic** [= *érték*]

Ez a tulajdonság **Form** és **MDIForm** objektumon, továbbá **Label**, **PictureBox** és **TextBox** vezérlőelemeken érvényes, és forrás űrlapon az általa egy DDE adatcserén keresztül visszaadott téma, míg a fogadó vezérlőelemen a forrás alkalmazás és téma (az alkalmazásban használt alapvető adatcsoportot) kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a DDE szintaxis elemét megadó karakteres kifejezés.

A tulajdonságban a karakterlánc a fogadó vagy forrás kapcsolat felépítéséhez szükséges információkat tartalmazza. A karakterlánc tartalma attól függ, hogy azt a forrás űrlaphoz vagy a fogadó vezérlőelemhez használjuk, de mindegyik a standard DDE szintaxis *alkalmazás*, *téma*, *elem* részeinek közül eggyel vagy többel van összhangban.

Megjegyzendő, hogy míg a standard definíció a DDE kapcsolathoz az *alkalmazás*, *téma* és *elem* elemeket tartalmazza, az alkalmazáson belül a fogadó kapcsolat forrás alkalmazásra irányuló aktuális szintaxisa ettől kissé eltérő lehet. Így például az Excelen belül az *alkalmazás*|*téma*!*elem*, az MS Word for Windowsban az *alkalmazás téma elem*,

míg a Visual Basicen belül az *alkalmazás|téma*; alak használható, ahol a ! jel a *téma* megadásába beleértődik.

A **LinkTopic** tulajdonság fogadó vezérlőelemhez történő beállításban a *kapcsolat* karakterlánc *alkalmazás|téma* alakban használandó:

- Az *alkalmazás* annak az alkalmazási űrlapnak a neve, amelytől adatok kérünk. Ez általában egy végrehajtható, kiterjesztés nélküli fájlnév, pl. EXCEL.
- Az alkalmazást és a témát a "|" jel (karakterkódja 124) választja el.
- A *téma* a forrás alkalmazásban használt alapvető adatcsoport, pl. egy Excel munkalapra való hivatkozás.

A csak fogadó vezérlőelemhez a kapcsolódó **LinkItem** tulajdonságot is be kell állítani a kapcsolatban az *elem* meghatározásához, például egy "R1C1" alakú cellahivatkozás az Excel munkalap egy elemének felel meg.

A **LinkTopic** tulajdonság forrás űrlaphoz történő beállításban az *érték* az űrlap egy megfelelő azonosítója. A továbbiakban a fogadó alkalmazás az űrlaphoz tartozó DDE kapcsolat létesítésekor ezt a karakterláncot fogja a *téma* argumentumként használni. Bár a Visual Basicben a forrás űrlaphoz csak ez a karakterlánc a **LinkTopic** tulajdonság egyetlen beállítandó eleme, a fogadó alkalmazás ugyancsak részletezést igényel:

- Ilyen az alkalmazási program által használt *alkalmazás* nevű elem, amely egy, az alkalmazási környezettől függően vagy egy Visual Basic projekt .VBP kiterjesztés nélküli fájlneve (az alkalmazás Visual Basic fejlesztői környezetben futtatásakor), vagy egy Visual Basic alkalmazás .EXE kiterjesztés nélküli fájlneve (az alkalmazás önálló futtatható fájlkénti futtatásakor) lehet. Ezt a karakterláncot — hacsak a fájlnevet a felhasználó nem változtatta meg — az **App** objektum **EXEName** tulajdonsága adja a Visual Basic program számára. (Az **App.EXEName** mindig az alkalmazás aktuális fájlnevet adja vissza, míg a DDE mindig a Make EXE párbeszédpanelen adott eredeti nevet használja.)
- Ugyancsak beállítandó a fogadó alkalmazás által használt *elem*, amelynek összhangban kell lennie a forrás űrlapon lévő **Label**, **PictureBox** vagy **TextBox** vezérlőelem **Name** tulajdonságával.

Az adatkapcsolat **LinkTopic** tulajdonsággal való aktiválásakor a **LinkMode** tulajdonságot a kívánt adatkapcsolati típust megadó, nullától különböző értékre kell beállítani. Általános szabály, hogy a **LinkMode** tulajdonság értéke a **LinkTopic** beállítása után adható meg.

Mivel a **LinkTopic** tulajdonságának megváltoztatása a fogadó vezérlőelemen a kapcsolatot megszakítja és a DDE adatcserét lezárja, a forrás űrlapon pedig az összes, ugyanazon témát használó fogadókapcsolatot megszakítja, a **LinkTopic** módosítása előtt a **LinkMode** tulajdonságot 0 értékre kell beállítani. A **LinkTopic** beállítása után a **LinkMode** tulajdonság értékét az új témával kívánt kapcsolati típusnak (1 — *Automatic*, 2 — *Manual*, 3 — *Notify*) megfelelően lehet beállítani.

A programfejlesztési fázisban az Edit menü Paste Link parancsával kialakított állandó adatkapcsolat a **LinkMode**, a **LinkTopic** és **LinkItem** tulajdonságot is beállítja. Ezt követően a kapcsolat az űrlapokkal együtt tárolódik, és az űrlap minden egyes betöltése után a Visual Basic megkísérli annak újbóli létrehozását.

List

Szintaxis: *objektum*.**List**(*index*) [= *érték*]

Ez a tulajdonság **ComboBox**, **DirListBox**, **DriveListBox**, **FileListBox** és **ListBox** vezérlőelemeken érvényes, és a vezérlőelemben a listamező elemeinek kiolvasására vagy beállítására használható. A programfejlesztési fázisban a **ComboBox** és **ListBox** vezérlőelemeken a tulajdonságtallózón keresztül használható, míg futásidőben csak olvasható a **DirListBox**, **DriveListBox**, **FileListBox** és írható és olvasható a **ComboBox** és **ListBox** vezérlőelemeken. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum a lista meghatározott elemének száma. A lista első eleme (az **Option Base** értékétől függetlenül) 0, az utolsó eleme pedig a **ListCount**-1 értékű lesz.
- A tulajdonság beállításához használható *érték* argumentum egy listaelemet meghatározó karakteres kifejezés.

A **DirListBox** vezérlőelemen a tulajdonság *index* argumentuma a vezérlőelem készítésekor aktuális könyvtár- és alkönyvtár-beosztáson alapuló sorszám. Az éppen érvényes könyvtárt az *index*-1, a felette lévő könyvtárszinteket negatív, a további alkönyvtárszinteket pedig nullától pozitív irányba növekvő, 0 és a **ListCount**-1 közötti értékek jelölik.

A **ComboBox** és **ListBox** vezérlőelemek listamezője kezdetben üres listát tartalmaz. A fájlrendszer vezérlőelemeinél a lista a vezérlőelem futásidőben történt létrehozásakor érvényes feltételeken alapul:

- A **DirListBox** vezérlőelemen a lista a *-n* és a **ListCount**-1 tartományon belüli összes könyvtárt tartalmazza.
- A **DriveListBox** a csatlakoztatott meghajtók listáját tartalmazza.
- A **FileListBox** az éppen érvényes szinten található, a **Pattern** tulajdonsággal szűrt fájlok (elérési út nélküli) listáját tartalmazza.

A **List** tulajdonság a **ListCount** és **ListIndex** tulajdonságokkal összhangban működik.

A **DirListBox** kivételével minden alkalmazható vezérlőelemnél az *index* 0 és **ListCount**-1 közötti értékeit sorba véve kapható meg a lista összes eleme. A **DirListBox** vezérlőelemnél az éppen aktuális szinten látható összes könyvtár és alkönyvtár listája az

index -n és **ListCount**-1 közötti értékeinek sorba vételével kapható meg. A *-n* az éppen aktuális feletti könyvtárszintek száma.

A **ComboBox** és **ListBox** vezérlőelemekben megjelenítendő elemek megadása az **AddItem** metódus használatával, míg elemek eltávolítása a **RemoveItem** metódussal lehetséges. A lista betűrendben történő megjelenítéséhez a vezérlőelem **Sorted** tulajdonságát a listához új elem csatolása előtt **True** értékre kell beállítani.

A Declaration részben használt `Option Base = 1` utasítás nem hat a Visual Basic vezérlőelemekben az elemek számozására, az első elem indexe mindig 0.

Ha az *index* az éppen aktuális listamezőben nem létező elemet jelöl, a tulajdonság üres karakterláncot ("") ad vissza.

ListCount

Szintaxis: *objektum*.**ListCount**

A tulajdonság **ComboBox, DirListBox, DriveListBox, FileListBox** és **ListBox** vezérlőelemeken érvényes, és a vezérlőelem listamezőjében szereplő elemek számát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **ListCount** tulajdonság értékének értelmezése az egyes vezérlőelemek esetén a következő:

- **ComboBox** és **ListBox** vezérlőelemeken a lista elemeinek száma
- **DirListBox** vezérlőelemen az aktuális könyvtár alkönyvtárainak száma
- **DriveListBox** vezérlőelemen a csatlakoztatott meghajtók száma
- **FileListBox** vezérlőelemen az aktuális könyvtárban található és a **Pattern** tulajdonságban adott mintának megfelelő fájlok száma

Ha nincs kiválasztott elem, a visszaadott érték -1. Egy listán a **ListIndex** = 0 jelöli ki az első elemet, és a **ListCount** tulajdonság értéke mindig eggyel nagyobb a legnagyobb **ListIndex** értéknél.

ListField

Szintaxis: *objektum*.**ListField** [= *érték*]

Ez a tulajdonság **DBCombo** és **DBList** vezérlőelemeken érvényes, és a **Recordset** objektumból a vezérlőelem listamezőjének feltöltéséhez használt mező nevének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **RowSource** tulaj-

donságban adott **Data** vezérlőelemmel létrehozott **Recordset** objektumbeli mező nevét megadó karakteres kifejezés.

A **ListField** tulajdonság lehetőséget ad egy vezérlőelem listamezőjébe egy **Recordset** objektum kívánt mezőjének betöltésére. A tulajdonság a lista feltöltésére használt **Recordset** létrehozásához használt **Data** vezérlőelemet meghatározó **RowSource** tulajdonsággal összhangban használható. Ha a **ListField** tulajdonság nincs megadva, a lista feltöltésére használt mezőt a **BoundColumn** tulajdonság határozza meg a **Recordset** objektumban. Ha sem a **ListField**, sem a **BoundColumn** tulajdonság sincs megadva, a lista üres lesz.

Általában két **Recordset** objektumot használunk az adatmegjelenítő listát tartalmazó vezérlőelemekkel. Az egyik az érvényes kiválasztás csak olvasható listáját tartalmazza, míg a másik a listakijelölésnek megfelelően frissített **Recordset**.

A **Recordset** objektumban nem található mezőt kijelölő **ListField** tulajdonság megadása kezelhető hibát okoz.

ListIndex

Szintaxis: *objektum*.**ListIndex** [= *index*]

Ez a programfejlesztési fázisban nem használható tulajdonság **ComboBox**, **DirListBox**, **DriveListBox**, **FileListBox** és **ListBox** vezérlőelemeken érvényes, és az adott vezérlőelem kiválasztott eleméhez tartozó index kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *index* argumentum az aktuális elem indexét megadó numerikus kifejezés.

Az *index* argumentum lehetséges értékei és értelmezésük a következő:

Ha az *index*:	Értelmezése:
-1	Nincs érvényes elem kiválasztva. (A **ComboBox**, **DirListBox** és **DriveListBox**, vezérlőelemeken alapértelmezett beállítás.) A **ComboBox** vezérlőelemen a szövegbeviteli mező be a felhasználó általi új szöveg beírását, a **DirListBox** vezérlőelemen az aktuális elérési út indexét, a **DriveListBox** vezérlőelemen pedig a futásidőben történt létrehozásakor érvényes meghajtó indexét jelzi.
n	Az aktuálisan kiválasztott elem indexének értéke. (A **FileListBox** és **ListBox** vezérlőelemeken alapértelmezett beállítás.)

A `List(Lista1.ListIndex)` kifejezés az éppen kiválasztott elem karakterláncát adja vissza.

Egy listán a **ListIndex** = 0 jelöli ki az első elemet, és a **ListCount** tulajdonság értéke mindig egyel nagyobb a legnagyobb **ListIndex** értéknél.

Az olyan vezérlőelemek esetén, amelyek többszörös választást is lehetővé tesznek, működésük a kiválasztott elemek számától függ. Ha csak egyetlen elem kiválasztott állapotú, a **ListIndex** tulajdonsággal meghatározható annak indexe, azonban több elem kiválasztottsága esetén a **ListIndex** a fókusz keretében lévő elem indexét adja vissza, függetlenül attól, hogy az éppen kiválasztott-e vagy sem.

Locked

Szintaxis: *objektum*.**Locked** [= *érték*]

Ez a tulajdonság **Column** objektumon és **TextBox** vezérlőelemen érvényes, és azok szerkeszthetőségét meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a szerkeszthetőséget meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True **Column** objektumon — Az oszlop értékei nem szerkeszthetők.
TextBox vezérlőelemen — A vezérlőelembeli szöveg görgethető és kijelölté tehető, de nem szerkeszthető. A szöveg a **Text** tulajdonsággal módosítható a programkódból.

False **Column** objektumon — Az oszlop értékei szerkeszthetők.
TextBox vezérlőelemen — A vezérlőelembeli szöveg szerkeszthető.

A **Column** objektum esetén a **Locked** tulajdonság alapértelmezett értéke a **DataUpdatable** tulajdonság értéke lesz az alapjául szolgáló mezőkön, ha azonban a **Column** nem kötött vagy az adatforrás nem támogatja a **DataUpdatable** tulajdonságot, az alapértelmezett érték **True** lesz. Ha a **DataUpdatable** tulajdonság értéke az alapjául szolgáló mezőkön **False**, a **Locked** tulajdonság **True** értékre változtatása nem okoz hibát. Hiba akkor következik be, ha a vezérlőelem a megváltoztatott értékeket megkísérli az adatbázisba írni.

LockEdits

Szintaxis: *objektum*.**LockEdits** [= *érték*]

A tulajdonság eredményhalmaz és tábla típusú **Recordset** (és az elavult **Dynaset** és **Table**) objektumon érvényes, és a szerkesztés során érvényes zárolási típust meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a követke-

ző objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a zárolási típust meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True Pesszimisztikus zárolás van érvényben. Az általunk szerkesztett rekordot tartalmazó 2K-s adatlap az **Edit** metódus használatakor azonnal zárolttá válik. A zárolás az **Update** vagy **Close** (illetve bármelyik **Move** vagy **Find**) metódus használatakor felszabadul. (Alapértelmezett.)

False Optimisztikus zárolás van érvényben. Az általunk szerkesztett rekordot tartalmazó 2K-s adatlap csak addig zárolt, amíg a rekordok frissítése az **Update** metódussal folyik.

Ha az adatlap zárolt (a **LockEdits** értéke **True**), más felhasználó nem tud ugyanazon lapon rekordokat szerkeszteni. A zárolás csak a szerkesztésre vonatkozik, a zárolt adatlapot más felhasználók olvashatják.

Ha a **LockEdits** értékének **True** beállítása olyankor történik, ha egy másik felhasználó már az adott lapot bezárta, az **Edit** metódus alkalmazása hibát okoz. Ugyancsak hibát okoz a **LockEdits** értékének **False** beállítása utáni **Update** metódus használata, ha ugyanakkor egy másik felhasználó zárolva tartja az adatlapot. A rekordunkon más felhasználó által végrehajtott változtatásokat a **Recordset** objektumunk **Bookmark** tulajdonságának sajátmagára történő beállításával ellenőrizhetjük.

Az ODBC adatforrásokkal végzett munka során a **LockEdits** értéke mindig **False**, azaz optimisztikus zárolás van érvényben. A Microsoft Jet adatbázis-szerkesztő nem ellenőrizheti a külső adatbázis-kiszolgálók által használt zárolási mechanizmusokat.

LoginTimeout

Szintaxis: *objektum*.**LoginTimeout** [= *érték*]

Ez a tulajdonság **DBEngine** objektumon érvényes, és egy ODBC adatbázisba történő belépés során a várakozási időkorlátot meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a várakozási idő túllépési hiba bekövetkeztéig terjedő, másodpercben értelmezett időt meghatározó **Integer** típusú szám. Az alapértelmezett érték 20 másodperc. 0 érték beállítása esetén nem kapunk időtúllépési hibajelzést.

Egy ODBC — például SQL Server — adatbázisba történő belépés hálózati hiba

vagy a kiszolgáló üzemen kívül állása miatt sikertelen lehet. A határozatlan ideig tartó várakozás helyett megadhatjuk a Microsoft Jet hibajelzés előtti várakozási idejét.

A Microsoft Jet számára a hibajelzés kiadása előtti alapértelmezett 20 másodperces várakozási idő helyett más időtartam is megadható. A **LoginTimeout** tulajdonságban aktuális beállítása az alkalmazás inicializálási (.INI) fájl [ODBC] részében határozza meg a várakozási időt.

LogMessages

Szintaxis: *objektum*.**LogMessages** [= *érték*]

Ez a felhasználó által definiált tulajdonság **QueryDef** objektumon érvényes, és az ODBC adatbázis visszatérési üzenetének rögzítettségét meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság és csak a létrehozáskor és telepítéskor írható és olvasható. A **LogMessages** tulajdonságot a felhasználónak kell a **QueryDef** objektumon a **CreateProperty** metódussal telepíteni, és csak ekkor írható és olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az üzenet rögzítettségét meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True	Van rögzítve ODBC által generált üzenet.
False	Nincs rögzítve ODBC által generált üzenet.

Egyes feldolgozásra átadott lekérdezések az adatok mellett üzeneteket is visszaadhatnak. Ha a **LogMessages** tulajdonság értéke **True**, a Microsoft Jet a visszaadott üzeneteket tartalmazó táblát hoz létre. A tábla neve *FELHASZNÁLÓNÉV-nn* alakú, ahol *nn* az üzenet 00 értékkel kezdődő sorszáma.

Ha olyan lekérdezésre számítunk, amely üzenetet ad vissza, a **QueryDef** objektumhoz hozzuk létre a **LogMessages** tulajdonságot, amelynek típusa legyen **Boolean** (**dbBoolean**) és értéke **True**.

A visszaadott üzenetek táblái feldolgozás után az adatbázisból az őket létrehozó ideiglenes lekérdezésekkel együtt törölhetők.

IpOleObject

Szintaxis: *objektum*.**IpOleObject**

A tulajdonság **OLE** befogadó vezérlőelemeken érvényes, és az objektum címét adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az OLE DLL-ekben számos függvényhívás egy objektum címének argumentumkénti átadását igényli. Az OLE DLL-ek API hívásaikor az **IpOleObject** tulajdonságban meghatározott értéket kell átadni. Ha nincs megjelenített objektum, a tulajdonság értéke 0. Olyan API hívásnál, amely az OLE vezérlőelemet visszahívja, az eredmény előre nem meghatározható lesz.

A tulajdonság által visszaadott cím az aktív objektum **IpOleObject** interfészre irányuló távoli mutató.

Major

Szintaxis: *objektum*.**Major**

Ez a futásidőben csak olvasható tulajdonság **App** objektumon érvényes, és a projekt 0 és 999 közötti értékű fő verziószámát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Major** tulajdonság a futó alkalmazás verziójára vonatkozó információt ad vissza. A tulajdonság beállítása a programfejlesztési fázisban az EXE Dialog párbeszédpanelen történik.

MatchEntry

Szintaxis: *objektum*.**MatchEntry** [= *érték*]

A tulajdonság **DBCombo** és **DBList** vezérlőelemeken érvényes, és a felhasználói bejegyzés alapján végzett keresés módját meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a fókuszban lévő vezérlőelembe egy vagy több karakter felhasználó általi beírásakor érvényes viselkedését meghatározó szimbolikus konstans vagy érték.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	A keresési mód:
vbMatchEntrySimple	0	*Basis Matching* — A keresés a lista elemeinek első karaktere és a beírt karakter egyezése szerint történik. Ugyanazon karakter ismétlődő beírásával a teljes lista végigkereshető. (Alapértelmezett.)

vbMatchEntryExtended	1	*Extended Matching* — A keresés a beírt összes karakter egyezése szerint történik. A művelet a karakterek beírásakor indul, később finomítva a keresést.

Ha a **MatchEntry** tulajdonság értéke **vbMatchEntryExtended** és a felhasználó leüti a BACKSPACE billentyűt vagy néhány másodpercet beírás nélkül várakozik, az illeszkedési mintát tartalmazó karakterlánc törlődik.

MatchWithList

Szintaxis: *objektum*.**MatchWithList**

A tulajdonság **DBCombo** és **DBList** vezérlőelemeken érvényes, és a listamező egy rekordjának és a **BoundText** tulajdonság értékének egyezését meghatározó értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **MatchWithList** tulajdonság lehetséges értékei és értelmezésük a következő:

True	A **BoundText** tulajdonság értéke megegyezik a lista egyik rekordjával.
False	A **BoundText** tulajdonság értéke nem egyezik a lista egyik rekordjával sem.

Amikor a **DBCombo** vezérlőelem szövegbeviteli mezőjébe egy értéket írunk, a **MatchWithList** tulajdonság **True** értékűvé válik, ha a beírt érték a listában szereplő elemek valamelyike. A tulajdonság használatával listában nem szereplő bejegyzéseket lehet kiszűrni vagy a forrástáblához új bejegyzéseket adó kódot lehet készíteni.

Max

Szintaxis: *objektum*.**Max** [= *érték*]

A **Max** tulajdonság a **CommonDialog**, valamint a **HScrollBar** és **VScrollBar** vezérlőelemeken eltérő értelmezéssel használható.

A **Max** tulajdonság a **CommonDialog** vezérlőelem Font párbeszédpaneljén a legnagyobb karakterméret, a Print párbeszédpaneljén pedig a legnagyobb nyomtatási tartomány érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a vezérlőelem Font párbeszédpaneljén a Size listamezőben megjelenő legnagyobb karakterméret pontban, a Print párbeszédpaneljén pedig a legnagyobb nyomtatási tartomány oldalszámban értelmezett értékét meghatározó numerikus kifejezés.

A **Max** használatával kapcsolatos további tudnivalók a következők:

- A tulajdonságnak Font párbeszédpanelen történő használata előtt a **LimitSize** jelzőnek beállítottnak kell lennie
- A tulajdonság a Print párbeszédpanelen a nyomtatási tartománynak a felhasználó által a To mezőben megadható maximális értékét határozza meg.

A **Max** tulajdonság a **HScrollBar** és **VScrollBar** vezérlőelemeken a **Value** tulajdonságnak a csúszka legfelső vagy jobb oldali legszélső helyzetében érvényes maximális értékének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **Value** tulajdonság maximális értékét meghatározó -32767 és az alapértelmezett 32767 közötti értékű, **Integer** típusú numerikus kifejezés.

A Windows operációs környezet a **Form** objektumon, valamint a **ComboBox** és **ListBox** vezérlőelemeken azok tartalmával arányos tartomány állít be, azonban a **HScrollBar** és **VScrollBar** vezérlőelemeken ezt a felhasználási követelményekkel összhangban, a **Max** és **Min** tulajdonságokkal magunknak kell elvégeznünk. Ha a **Max** tulajdonság értéke kisebb, mint a **Min** értéke, a gördítősáv iránya megfordul.

A tulajdonság beállítása jellemzően a programfejlesztési fázisban történik, azonban lehetséges a futásidőben történő dinamikus beállítás is.

A gördítési lépésnagyság a **LargeChange** és a **SmallChange** tulajdonságok beállításaival határozható meg.

MaxButton

Szintaxis: *objektum*.**MaxButton**

A tulajdonság **Form** objektumon érvényes, és az űrlapon a Maximize gomb meglétét meghatározó értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **MaxButton** tulajdonság lehetséges értékei és értelmezésük a következő:

True	Az űrlapon van Maximize gomb. (Alapértelmezett.)
False	Az űrlapon nincs Maximize gomb.

A felhasználó a Maximize gomb használatával teheti az űrlap ablakát teljes képernyőméretűvé. A Maximize gomb megjelenítéséhez az űrlap **BorderStyle** tulajdonságát 1 (*Fixed Single*) vagy 2 (*Sizable*) értékre kell beállítani. A Maximize gomb az űrlap teljes méretűvé válása után automatikusan Restore, azaz az űrlapot eredeti méretre visszaállító gombbá válik, a visszaállítás után pedig ismét az eredeti Maximize funkciójú gomb lesz.

A **MaxButton**, **MinButton**, **BorderStyle** és **ControlBox** tulajdonságok beállításai csak futásidőben hatnak az űrlap megjelenésére.

Az űrlap futásidőben történő teljes méretűvé tétele egy **Resize** eseményt eredményez. Az ablak aktuális állapotát a **WindowState** tulajdonság tükrözi: ha értéke 2 (*Maximized*), az űrlap a **MaxButton** és **BorderStyle** értékétől függetlenül teljes méretű lesz.

MaxFileSize

Szintaxis: *objektum*.**MaxFileSize** [= *érték*]

A tulajdonság **CommonDialog** vezérlőelemen (File párbeszédpanel) érvényes, és a vezérlőelem használatával megnyitott maximális fájl nagyságának kiolvasására vagy a megnyitható maximális fájlnagyság beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a maximális megnyitható, 1 és 32K közötti fájlnagyság értékét meghatározó **Integer** érték. Az alapértelmezett *érték* 256 byte.

MaxLength

Szintaxis: *objektum*.**MaxLength** [= *érték*]

A tulajdonság **TextBox** vezérlőelemen érvényes, és a vezérlőelemben elhelyezhető karakterek számának korlátozását jelző, illetve a maximális karakterszámot meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a felhasználó által a **TextBox** vezérlőelembe írható karakterek maximális számát meghatározó **Integer** típusú érték. Az *érték* alapértelmezetten 0, amely az egysoros **TextBox** esetén a felhasználó rendszerében a memória nagysága által korlátozott, többsoros **TextBox** esetén pedig kb. 32K-s hosszúságkorlát érvényes. Bármely, ettől eltérő *érték* maximális karakterszámként lesz értelmezett.

A **MaxLength** tulajdonság a **TextBox** vezérlőelembe beírható karakterek számának korlátozására használható.

A futásidőben beállított **MaxLength** értékét meghaladó hosszúságú szöveg beírása nem okoz hibát, azonban csak a beállított számú karakter kerül a **Text** tulajdonságba, a túlnyúló rész elvész. A tulajdonság módosítása a szövegbeviteli mező aktuális tartalmára nincs hatással, csak a módosítást követően változások esetén érvényesül.

MDIChild

Szintaxis: *objektum*.**MDIChild**

Ez a futásidőben csak olvasható tulajdonság **Form** objektumon érvényes, és az az adott űrlapnak egy MDI-űrlapon belül MDI-alűrlapkénti megjelenését meghatározó értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **MDIChild** tulajdonság lehetséges értékei és értelmezésük a következő:

True	Az űrlap egy MDI-alűrlap és az MDI-űrlapon belül jelenik meg.
False	Az űrlap nem egy MDI-alűrlap.(Alapértelmezett.)

A tulajdonságot MDI-alkalmazások létrehozásakor lehet használni. Futásidőben az az űrlap, amely **MDIChild** tulajdonsága **True**, egy MDI-űrlapon belül jelenik meg. Az MDI-alűrlap a főűrlapon belül teljes méretűvé tehető, ikonállapotba hozható és mozgatható.

Az MDI-alűrlapok használatával kapcsolatban a következőkre kell figyelemmel lenni:

- Futásidőben a teljes méretű MDI-alűrlap címe a címsorban a főűrlap címével kombinálva jelenik meg.
- A programfejlesztési fázisban az MDI-alűrlap más űrlapokhoz hasonlóan jelenik meg, csak futásidőben kerül a főűrlapon belülre. Az MDI-alűrlap ikonja a Projekt ablakban az egyéb űrlapok ikonjaitól eltérő lesz.
- Az MDI-alűrlap nem lehet modális.
- Az MDI-alűrlap kiindulási mérete és helyzete a Windows operációs környezettől függő lesz, kivéve ha azokat a **Load** eseményeljárásban beállítottuk.
- Ha egy MDI-alűrlapra azelőtt történik hivatkozás, hogy a hozzá tartozó főűrlap betöltött lenne, a főűrlap is automatikusan betöltődik, azonban ez fordítva már nem működik.

A Windows 3.0 vagy későbbi verzió alatti futás során minden MDI-alűrlap méretezhető kerettel, Control menüvel, továbbá Minimize és Maximize gombokkal jelenik meg, függetlenül a **BorderStyle**, **ControlBox**, **MinButton** és **MaxButton** tulajdonságainak beállításától.

Bármely **MDIForm** objektumra történő hivatkozás — beleértve tulajdonságainak beállítását és kiolvasását is — betölti és láthatóvá teszi az adott űrlapot.

Min

Szintaxis: *objektum*.**Min** [= *érték*]

A tulajdonság a **CommonDialog**, valamint a **HScrollBar** és **VScrollBar** vezérlőelemeken eltérő értelmezéssel használható.

A **Min** tulajdonság a **CommonDialog** vezérlőelem Font párbeszédpaneljén a legkisebb karakterméret, a Print párbeszédpaneljén pedig a legkisebb nyomtatási tartomány érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a vezérlőelem Font párbeszédpaneljén a Size listamezőben megjelenő legkisebb karakterméret pontban, a Print párbeszédpaneljén pedig a legkisebb nyomtatási tartomány oldalszámban értelmezett értékét meghatározó numerikus kifejezés.

A **Min** használatával kapcsolatos további tudnivalók a következők:

- A tulajdonságnak Font párbeszédpanelen történő használata előtt a **LimitSize** jelzőnek beállítottnak kell lennie
- A tulajdonság a Print párbeszédpanelen a nyomtatási tartománynak a felhasználó által a From mezőben megadható maximális értékét határozza meg.

A **Min** tulajdonság a **HScrollBar** és **VScrollBar** vezérlőelemeken a **Value** tulajdonságnak a csúszka legalsó vagy bal oldali legszélső helyzetében érvényes minimális értékének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **Value** tulajdonság minimális értékét meghatározó -32767 és az 32767 közötti (alapértelmezetten 0) értékű, **Integer** típusú numerikus kifejezés.

A tulajdonsággal kapcsolatos egyéb tudnivalók értelemszerűen azonosak a **Max** tulajdonságnál már leírtakkal.

MinButton

Szintaxis: *objektum*.**MinButton**

A tulajdonság **Form** objektumon érvényes, és az űrlapon a Minimize gomb meglétét meghatározó értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **MinButton** tulajdonság lehetséges értékei és értelmezésük a következő:

True	Az űrlapon van Minimize gomb. (Alapértelmezett.)
False	Az űrlapon nincs Minimize gomb.

A felhasználó a Minimize gomb használatával az űrlap ablakát ikonná zsugoríthatja. A Minimize gomb megjelenítéséhez az űrlap **BorderStyle** tulajdonságát 1 (*Fixed Single*) vagy 2 (*Sizable*) értékre kell beállítani.

A **MaxButton**, **MinButton**, **BorderStyle** és **ControlBox** tulajdonságok beállításai csak futásidőben hatnak az űrlap megjelenésére.

Az űrlap futásidőben történő ikonméretűvé tétele egy **Resize** eseményt eredményez. Az ablak aktuális állapotát a **WindowState** tulajdonság tükrözi: ha értéke 2 (*Maximized*), az űrlap a **MaxButton** és **BorderStyle** értékétől függetlenül teljes méretű lesz.

Minor

Szintaxis: *objektum*.**Minor**

Ez a futásidőben csak olvasható tulajdonság **App** objektumon érvényes, és a projekt 0 és 999 közötti értékű al-verziószámát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Minor** tulajdonság a futó alkalmazás verziójára vonatkozó információt ad vissza. A tulajdonság beállítása a programfejlesztési fázisban az EXE Dialog párbeszédpanelen történik.

MiscFlags

Szintaxis: *objektum*.**MiscFlags** [= *érték*]

A tulajdonság **OLE** befogadó vezérlőelemen érvényes, és a vezérlőelem egy vagy több járulékos viselkedéséhez való hozzáférést meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy járulékos viselkedéséhez való hozzáférést meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentumhoz tartozó beállítások a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	A hozzáférés:
vbOLEMiscFlagMemStorage	0	A vezérlőelem a betöltött objektumot a memóriában tárolja.
vbOLEMiscFlagDisableInPlace	1	Felülírja a vezérlőelem alapértelmezett viselkedésmódját, lehetővé téve az azt támogató objektumok helybeni akivitását.

A **MiscFlags** tulajdonság **bOLEMiscFlagMemStorage** jelzőjének beállítása az objektum alapértelmezettnél gyorsabb működését eredményezi az objektum időleges fájlkénti lemezen történő tárolásának elmaradása következtében, azonban a nagy hely-

igényű adatokat használó objektumok miatt nagy memóriaterületet használ.

Ha az objektum támogatja a helybeni aktivitást, az objektumot a **vbOLEMiscFlagDisableInPlace** beállítással lehet egy külön ablakban történő aktivitása kényszeríteni.

A szimbolikus konstansok egymással kombinálva is használhatók az **Or** operátor alkalmazásával.

MouseIcon

Szintaxis 1: *objektum*.**MouseIcon** = **LoadPicture**(*elérési_út*)

Szintaxis 2: *objektum*.**MouseIcon** = *kép*

A tulajdonság **Form**, **MDIForm** és **Screen** objektumokon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **Data**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox** vagy **VScrollBar** vezérlőelemeken érvényes, és egy testre szabott egérikon beállítására használható. A tulajdonság szintaxisai a következő objektumazonosítót és argumentumokat tartalmazzák:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *elérési_út* argumentum a testre szabott ikont tartalmazó fájl nevét és elérési útját tartalmazó karakteres kifejezés.
- A *kép* argumentum egy **Form** objektum, továbbá **PictureBox** vagy **Image** vezérlőelem **Picture** tulajdonsága lehet.

A **MouseIcon** tulajdonság egy testre szabott ikont szolgáltat, ha a **MousePointer** tulajdonságnak 99 értéket adunk.

Bár a Visual Basic 4.0-s verziója nem hoz létre mutatófájlokat (.CUR), a **MouseIcon** használatával mutató- vagy ikonfájlokat lehet betölteni, így a programkódból tetszés szerint nagyságú és érzékelőpont-helyzetű, testre szabott mutatóhoz lehet jutni. A Visual Basic 32-bites verziója nem tölti be az animált mutatófájlokat (.ANI), noha a Windows 32-bites verziója támogatja azokat.

MousePointer

Szintaxis: *objektum*.**MousePointer** [= *érték*]

A tulajdonság **Form**, **MDIForm** és **Screen** objektumokon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **Data**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox** vagy **VScrollBar** vezérlőelemeken érvényes, és a futásidőben egy objektum felett lévő egérmutató esetén megjelenő testre szabott egérikon típusát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság

szintaxisai a következő objektumazonosítót és argumentumot tartalmazzák:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a megjelenő egérmutató típusát meghatározó **Integer** érték.

Az *érték* argumentum értékétől függő egérmutató választéka a következő:

Ha az *érték*:	A mutató típusa:
0	Az objektum által meghatározott alak. (Alapértelmezett.)
1	Nyíl
2	Hajszálkereszt
3	I-alak
4	Ikon (egy kereten belüli kis keret)
5	Négy nyíl keresztalakban (észak, dél, kelet és nyugat irányban)
6	Két ellentétes irányú nyíl (északkelet és délnyugat irányban)
7	Két ellentétes irányú nyíl (észak és dél irányban)
8	Két ellentétes irányú nyíl (északnyugat és délkelet irányban)
9	Két ellentétes irányú nyíl (kelet és nyugat irányban)
10	Felfelé mutató nyíl
11	Homokóra
12	Nincs leejtés mutató
13	Nyíl és homokóra
14	Nyíl és kérdőjel
15	Minden irányú (Microsoft Windows NT 3.51)
99	A **MouseIcon** tulajdonságban meghatározott testre szabott ikon.

A tulajdonságot a működésmód változásának jelzésére használhatjuk, például az űrlap vagy egy párbeszédpanel feletti egérmutató esetén, míg a homokóra arra használható, hogy a felhasználót egy művelet befejeződésére való várakozás szükségességéről tájékoztassuk.

A **Screen** objektumra beállított **MousePointer** tulajdonság esetén az egérmutató a teljes képernyőn változhat. Ha az alkalmazásunk nem hívja a **DoEvents( )** függvényt és nem 32-bites, a **MousePointer** tulajdonság a teljes képernyőn található minden vezérlőelem és más alkalmazás **MousePointer** tulajdonságát felülírja. Ha az alkalmazás meghívja a **DoEvents( )** függvényt, a **MousePointer** tulajdonság egy testre szabott vezérlőelem felett időlegesen változhat.

MultiLine

Szintaxis: *objektum*.**MultiLine**

Ez a futásidőben csak olvasható tulajdonság **TextBox** vezérlőelemen érvényes, és a többsoros szöveg elfogadását és megjelenítését meghatározó értéket ad vissza. A tulaj-

donság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **MultiLine** tulajdonság lehetséges értékei és értelmezésük a következő:

True A szövegbeviteli mező többsoros szöveget is elfogad, a beíráskor a sorhosszon túlnyúló részt automatikusan a következő sorba helyezve.

False Figyelmen kívül hagyja a kocsivissza karaktert és a beírható szöveget egyetlen sornyira korlátozza. (Alapértelmezett.)

A nagyobb méretű többsoros szövegekben való munka megkönnyítésére az ilyen **TextBox** vezérlőelemeket gördítősávokkal lehet kiegészíteni. Ha nem használunk vízszintes gördítősávot (**HScrollBar** vezérlőelemet), a többsoros **TextBox** vezérlőelembe beírt szöveg sorokba tördelése automatikusan történik.

A fókuszt a többsoros **TextBox** vezérlőelem következő sorára az alapértelmezett parancsgombot nem tartalmazó űrlapoknál az ENTER billentyű, az alapértelmezett parancsgombot tartalmazó űrlapoknál pedig a CTRL+ENTER billentyűk leütésével lehet átlépni.

MultiSelect

Szintaxis: *objektum*.**MultiSelect**

Ez a futásidőben csak olvasható tulajdonság **FileListBox** és **ListBox** vezérlőelemen érvényes, és a többszörös kiválasztás lehetőségét és módját meghatározó értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **MultiSelect** tulajdonság lehetséges értékei és értelmezésük a következő:

Ha a **MultiSelect**:	Értelmezése:
0	Egyidejűleg több elem nem lehet kiválasztott. (Alapértelmezett.)
1	Egyszerű többszörös kiválasztás megengedett: a kiválasztás vagy a kiválasztás érvénytelenítése a SZÓKÖZ billentyűvel vagy egy egérkattintással lehetséges. A fókusz mozgatása a mutatóbillentyűkkel történhet.
2	Bővített többszörös kiválasztás megengedett: a SHIFT+egérkattintással vagy SHIFT+mutatónyíl (FEL, LE, JOBB, BAL) használatával az előzőleg kiválasztott elemtől az aktuális elemig terjedő minden elem kiválasztottá válik. Egy elemet CTRL+egérkattintás használatával lehet kiválasztottá tenni vagy kiválasztását érvényteleníteni.

Name

Szintaxis 1: *objektum*.**Name** [= *érték*]

Szintaxis 2: *objektum*.**Name**

Ezt a tulajdonságot az adathozzáférési objektumokon, valamint az általános objektumokon és vezérlőelemeken eltérő szintaxissal és értelmezéssel lehet használni.

A tulajdonság a Szintaxis 1 szerint **Column**, **Container**, **Database**, **DBEngine**, **Field**, **Group**, **Index**, **Parameter**, **Property**, **QueryDef**, **Recordset**, **TableDef**, **Workspace** és **User** (valamint az elavult **Dynaset**, **Snapshot** és **Table**) objektumokon érvényes, és egy adathozzáférési objektum felhasználó által definiált nevének kiolvasására vagy beállítására használható. Objektumgyűjteményhez még nem tartozó objektumoknál a **Name** írható és olvasható is. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a nevet meghatározó **String** típusú kifejezés. A névnek betűvel kell kezdődnie és legfeljebb 40 karaktert tartalmazhat, közötte számokat és az aláhúzás (_) karakter is, azonban írásjeleket és szóközt nem. A **TableDef** és **QueryDef**, valamint a **User** és a **Group** objektumok nem használhatják ugyanazon nevet.

Egy SQL utasítás használatával megnyitott **Recordset** objektum **Name** tulajdonsága az SQL utasítás első 256 karaktere.

A programkódban egy objektum **Name** tulajdonságát a **Dim** utasítással együtt az objektum másik példányának létrehozására használhatjuk.

A **Database** objektumok legtöbbjénél a **Name** a **Database** által ismert nevet tükrözi, és nincs kapcsolat a **Database** objektum neve és az objektumra való hivatkozáshoz használt objektumváltozó között.

Objektumgyűjteményhez még nem tartozó objektumnál a **Name** tulajdonság írhatósága és olvashatósága az objektum típusától a következőképpen függ:

Ha az objektum:	A tulajdonság használata:
Container	Csak olvasható.
Database	Csak olvasható.
Document	Csak olvasható.
Field	Írható és olvasható.
Index.Field	Csak olvasható.
QueryDef.Field	Csak olvasható.
Recordset.Field	Csak olvasható.
TableDef.Field (eredeti)	Írható és olvasható.
TableDef.Field (csatolt)	Csak olvasható.
Relation.Field	Csak olvasható.

Group	Írható és olvasható.
Index	Írható és olvasható.
Recordset.Index	Írható és olvasható.
Parameter	Csak olvasható.
Property (beépített)	Csak olvasható.
Property (felhasználói)	Írható és olvasható.
QueryDef	Írható és olvasható.
Recordset	Csak olvasható.
Relation	Írható és olvasható.
TableDef (eredeti)	Írható és olvasható.
TableDef (csatolt)	Írható és olvasható.
User	Írható és olvasható.
Workspace	Írható és olvasható.

A tulajdonság a Szintaxis 2 szerint **Application**, **Component**, **Database**, **Font**, **Form**, **MDIForm** és **Property** objektumokon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **CommonDialog**, **Data**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **Line**, **ListBox**, **Menu**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox**, **Timer** és **VScrollBar** vezérlőelemeken érvényes, és egy űrlap, vezérlőelem vagy objektum nevét adja vissza. A szintaxisban az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés. Ha ez hiányzik, az éppen fókuszban lévő **Form** válik az alapértelmezett objektummá.

Az új objektumok alapértelmezett neve az objektum sorszámmal kibővített fajtája, például `Form1` vagy `TextBox4`. A **Name** tulajdonságban érvényes névnek betűvel kell kezdődnie és legfeljebb 40 karaktert tartalmazhat, közötte számokat és az aláhúzás (_) karakter is, azonban írásjeleket és szóközt nem.

Az űrlap neve nem lehet más publikus érvényességű, például **Clipboard**, **Screen** vagy **App** objektuméval azonos. Noha — eltérően a Visual Basic 1.0 verziótól — a név kulcsszavakkal, tulajdonságok vagy más objektumok nevével is azonos lehet, az ilyen névadást kerülni kell a programkódban esetlegesen fellépő ütközések elkerülésére.

A programkódban egy űrlap **Name** tulajdonságát a **Dim** utasítással együtt az űrlap egy másik példányának létrehozására használhatjuk. A programfejlesztési fázisban nem lehet két azonos nevű űrlap.

Azonos típusú vezérlőelemek számára azonos **Name** tulajdonságot beállítva vezérlőelemtömbje hozható létre, amelyen belül az egyes elemeket azok **Index** tulajdonságukkal lehet egymástól megkülönbözteti.

A Visual Basic beépülőkkel megjelenített vagy átadott **Application** objektum esetén a **Name** tulajdonság beállítása mindig `Microsoft Visual Basic`, a **Property** objektum esetén pedig a Visual Basic objektumtárban az Object Browser által adott listában szereplő név lesz.

Bár a Visual Basic a **Name** tulajdonság beállítását gyakran a **Caption**, **LinkTopic**

és **Text** tulajdonságok alapértelmezett értékeként használja, ezek egyikének módosítása nem hat a többiekre.

Negotiate

Szintaxis:

Ez a futásidőben nem elérhető tulajdonság a **Data** és **PictureBox** objektumon érvényes, és használatával az állítható be, hogy egy pozícionálható vezérlőelem megjelenjene, amikor egy objektum az űrlapon egy vagy több eszköztárat jelenít meg.

A tulajdonság lehetséges értékei a következők:

True Az űrlapon belül pozícionálható vezérlőelem (az **Align** tulajdonság értéke nem 0) akkor is látható marad, amikor az űrlap egy aktív objektuma egy eszköztárat jelenít meg.

False Az űrlapon belül pozícionálható vezérlőelem nem marad látható, amikor az űrlap egy aktív objektuma egy eszköztárat jelenít meg. Az eszköztár a vezérlőelem helyén jelenik meg. (Alapértelmezett.)

A **Negotiate** minden **Align** tulajdonsággal is rendelkező vezérlőelemen létező tulajdonság. Az **Align** tulajdonság egy vezérlőelemnek **Form** vagy **MDIForm** objektumon belüli pozícionálására használható, azonban az eszköztárcsere csak **MDIForm** objektumon történhet. A pozícionált vezérlőelemnek **MDIForm** objektumon kell lennie.

Ha a **NegotiateToolbars** tulajdonság értéke **False**, a **Negotiate** tulajdonság hatástalan lesz.

NegotiateMenus

Szintaxis:

Ez a futásidőben nem elérhető tulajdonság a **Form** objektumon és **Forms** objektumgyűjteményen érvényes, és használatával egy azt meghatározó érték állítható be, hogy az űrlap menüsorában tartalmazza-e egy objektumának menüit.

A tulajdonság lehetséges értékei a következők:

True Amikor az űrlap egy objektuma szerkesztésre aktív, az objektum menüi megjelennek az űrlap menüsorában. (Alapértelmezett.)

False Az űrlap objektumainak menüi nem jelennek meg az űrlap menüsorában.

A **NegotiateMenus** tulajdonság használatával meghatározható, hogy egy űrlap menüsora a területét megosztva használja-e az űrlap aktív objektumának menüivel. Ha nem kívánjuk, hogy az űrlapunk menüsora tartalmazza az aktív objektum menüit, a **NegotiateMenus** tulajdonságot **False** értékre kell beállítani.

Ha a **NegotiateMenus** tulajdonság **True**, az űrlapnak menüsorral — akár rejtettel is

— kell rendelkeznie. Ebben a beállításban az egyedi **Menu** vezérlőelemek **NegotiatePosition** tulajdonságát lehet használni azon menük kijelölésére, amelyeket az űrlap az aktív objektum menüivel együtt jelenít meg.

Ha a **NegotiateMenus** tulajdonság értéke **False**, a **NegotiatePosition** tulajdonság hatástalan lesz.

A menüket nem lehet egy **MDIForm** objektum és az **MDIForm** egy objektuma között megosztani.

Ha az űrlap **MDIChild** tulajdonsága **True**, az aktív objektum menüi megjelennek az MDI-főablak, azaz az **MDIForm** objektum menüsorában.

NegotiatePosition

Szintaxis:

Ez a futásidőben nem elérhető tulajdonság a **Menu** vezérlőelemen érvényes, és használatával egy azt meghatározó érték állítható be, hogy amíg egy űrlapon egy csatolt vagy beágyazott objektum aktív és megjeleníti a menüit, a legfelső szintű **Menu** vezérlőelemek a menüsorban megjelenítetek legyenek-e.

A tulajdonság lehetséges értékei a következők:

Ha a **NegotiatePosition**:	Értelmezése:
0	*None* — Amíg az objektum aktív, a menü nem jelenik meg a menüsorban. (Alapértelmezett.)
1	*Left* — Amíg az objektum aktív, a menü a menüsor bal szélén jelenik meg a menüsorban.
2	*Middle* — Amíg az objektum aktív, a menü a menüsor közepén jelenik meg a menüsorban.
3	*Right* — Amíg az objektum aktív, a menü a menüsor jobb szélén jelenik meg a menüsorban.

A **NegotiatePosition** tulajdonság használatával az űrlapunk menüsorán azok az egyedi menük határozhatók meg, amelyek az űrlap egy aktív objektumával megosztva használják a menüsor területét.

Minden menü, amely **NegotiatePosition** tulajdonságának értéke nem 0, az aktív objektum menüivel együtt jelenik meg az űrlap menüsorában.

Ha a **NegotiateMenus** tulajdonság értéke **False**, a **NegotiatePosition** tulajdonság hatástalan lesz.

NegotiateToolbars

Szintaxis:

Ez a futásidőben nem elérhető tulajdonság a **MDIForm** objektumon érvényes, és

használatával egy azt meghatározó érték állítható be, hogy amikor az objektum az MDI-alablakon aktív, egy MDI-alablak objektumának eszköztára megjelenített legyen-e az **MDIForm** objektumon.

A tulajdonság lehetséges értékei a következők:

True Az **MDIForm** objektum megjeleníti az aktív objektum eszköztárait. Az eszköztáraknak az **MDIForm** objektum tetején vagy alján történő megjelenése az aktív objektumtól függ. (Alapértelmezett.)

False Az aktív objektum eszköztárait egyáltalán nem, vagy az aktív objektumtól függő lebegő eszközpalettaként jelennek meg.

A **NegotiateToolbars** tulajdonság az MDI-alablakokon objektumokat tartalmazó MDI-alkalmazás létrehozásakor használható. A tulajdonsággal az határozható meg, hogy az aktív objektum hogyan jelenítse meg az eszköztárait. A tulajdonság **True** értéke mellett az **MDIForm** az űrlap alsó vagy felső területét az aktív objektum eszköztárainak megjelenítésére megosztja.

Ha az **MDIForm** ugyancsak tartalmaz eszköztárat, a **Negotiate** tulajdonsággal írható elő, hogy a rendelkezésre álló területet az eszköztárak hogyan osszák meg egymás között.

NewIndex

Szintaxis: *objektum*.**MultiSelect**

Ez a tulajdonság **ComboBox** és **ListBox** vezérlőelemen érvényes, és a vezérlőelemhez legutolsóként adott elem indexét adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság elsősorban sorbarendezett listáknál lehet jól használni, amikor az **ItemData** tulajdonságtömb minden egyes elemének megfelelő értéklistára van szükség. Amint a listához egy új elemet adunk, azt a Visual Basic a listába betűrend alapján helyezi el. A **NewItem** tulajdonság megmondja, hogy az elem hová került, így az **ItemData** tulajdonságba ugyanazon indexszel lehet egy megfelelő értéket elhelyezni.

A **NewIndex** -1 értéket ad vissza, ha nincs a listában elem, vagy ha egy elem az utolsó elem hozzáadása óta törölve lett.

NoMatch

Szintaxis: *objektum*.**NoMatch**

A tulajdonság **Recordset** (valamint az elavult **Table**, **Dynaset** és **Snapshot**) objektumon használható, és egy rekordnak valamelyik **Find** (**FindFirst**, **FindPrevious**, **FindNext** vagy **FindLast**) vagy a **Seek** metódussal történő keresés sikerességét jelző értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot ki-

jelölő objektumkifejezés.

A tulajdonság lehetséges értékei a következők:

True Valamelyik **Find** metódus vagy a **Seek** metódus sikertelen volt, a keresett rekord nem volt található.

False Valamelyik **Find** metódus vagy a **Seek** metódus sikeresen megtalálta a kívánt rekordot.

A **Recordset** objektum **NoMatch** tulajdonság értéke a létrehozásakor vagy megnyitásakor **False** lesz.

Egy rekord keresése egy tábla típusú **Recordset** (valamint az elavult **Table**) objektumon a **Seek** metódussal, egy eredményhalmaz vagy pillanatfelvétel típusú **Recordset** (valamint az elavult **Dynaset** és **Snapshot**) objektumon valamelyik **Find** metódussal lehetséges. A keresés sikerességét a **NoMatch** tulajdonság értéke jelzi.

Ha a **Seek** metódus sikertelen és a **NoMatch** értéke **True**, az aktuális rekord a továbbiakban nem lesz érvényes.

Ha valamelyik **Find** (**FindFirst**, **FindPrevious**, **FindNext** vagy **FindLast**) metódus sikertelen és a **NoMatch** értéke **True**, az aktuális rekord a **Recordset** első rekordja lesz.

Ha a továbbiakban nem kívánunk az adott rekordra visszatérni, a **Seek** vagy bármelyik **Find** metódus használata szükséges az aktuális rekord könyvjelzőjének meghatározása.

A **Recordset** objektumon bármelyik **Find** metódus használata hatástalan lesz a **NoMatch** tulajdonság értékére.

Normal

Szintaxis: *objektum*.**Normal** [= *érték*]

A **Normal** tulajdonság **FileListBox** vezérlőelemeken érvényes, és a vezérlőelemben listázandó fájlok **Normal** attribútumát beállító érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A **FileListBox** vezérlőelemben a **Normal** attribútumú fájlokat jeleníti meg. (Alapértelmezett)

False A **FileListBox** vezérlőelemben a nem **Normal** attribútumú fájlokat jeleníti meg.

A tulajdonsággal a **FileListBox** vezérlőelemben megjelenítendő, az operációs kör-

nyezetben használt standard fájlattribútum állítható be. Ha a tulajdonság beállítása futásidőben, programkódból történik, a **FileListBox** vezérlőelem újra beállítódik és azon csak az adott attribútumú fájlok jelennek meg.

Number

Szintaxis: *objektum*.**Number** [= *hibaszám*]

A tulajdonság **Error** és **Err** objektumokon eltérő módon érvényes.

Az **Error** objektumon a **Number** alapértelmezett tulajdonság, és egy hibát meghatározó, **Long** típusú numerikus értéket ad vissza.

A **Number** tulajdonságot egy bekövetkezett hiba meghatározására lehet használni. A visszakapott érték egy hibafeltételnek megfelelő egyedi hibaértéknek felel meg. Az összes hibaszámot és hibafeltételt a Trappable Data Access Errors listája tartalmazza.

Az **Err** objektumon a **Number** alapértelmezett tulajdonság, és egy hibát meghatározó numerikus érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisai a következő objektumazonosítót és argumentumot tartalmazzák:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *hibaszám* argumentum egy Visual Basic hibaszámot vagy egy OLE SCode hibaértéket meghatározó **Long** érték.

Mivel a **Number** az **Err** objektum alapértelmezett tulajdonsága, az **Err( )** függvényt használó meglévő programkódot szükségtelen módosítani.

Egy OLE Automatizmus objektumból felhasználó által definiált hiba visszaadásához az **Err.Number** értékét a **vbObjectError** konstanshoz egy választott hibaszám hozzáadásával lehet beállítani.

Nem mindegyik Visual Basic kiszolgáló alkalmazás hozhat létre OLE objektumokat (ezt a kiszolgáló dokumentációjából lehet megtudni), azonban mindegyik létrehozhat OLE Automatizálási vezérlőelemeket, amelyeket a hibakódok értelmezését végző programkódokban lehet használni.

Mivel a **Number** tulajdonság, egy függvényhíváshoz hasonlóan viselkedik, ami lényeges lehet, ha az **Err** értéke egy eljárás számára van átadva. Ha egy meglévő programkódban az **Err** változását egy eljárás paramétereként adja vissza a hívó számára, a működés hibás lehet, ha az **Err** átadása hivatkozással (**ByRef**) történik. Azonban, mivel az **Err** globális objektum, argumentumkénti átadása szükségtelen, a hívott eljárásban az **Err** vagy **Err.Number** közvetlenül módosítható.

NumberFormat

Szintaxis: *objektum*.**NumberFormat** [= *érték*]

A tulajdonság **Column** objektumon érvényes, és egy **DBGrid** vezérlőelem **Column**

objektumának formáját meghatározó karakterláncot jelölő érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **Value** tulajdonságban a kifejezés formázását megadó karakteres kifejezés.

A **Text** tulajdonság ebből, a **Value** tulajdonságra alkalmazott formából származik. Ha a **NumberFormat** értéke érvénytelen karakterlánc, a cellákban lévő adatok #ERR# alakban jelenik meg, a **Value** tulajdonság értéke pedig változatlan marad. A forma számára érvényes karakterláncok a **Format( )** függvény ismertetésénél találhatók.

Object

Szintaxis 1: *objektum*.**Object**[.*tulajdonság* | .*metódus*]

Szintaxis 2: *objektum*.**Object**[.*tulajdonság* | .*metódus*] [= *érték*]

Ezt a tulajdonságot az **OLE** befogadó vezérlőelemen, valamint **CommonDialog** és **Grid** vezérlőelemeken eltérő szintaxissal és értelmezéssel lehet használni.

Ez a tulajdonság a Szintaxis 1 szerint **OLE** befogadó vezérlőelemen érvényes, és egy **OLE** befogadó vezérlőelembeli objektumot és/vagy objektumtulajdonságot ad vissza. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *tulajdonság* argumentum az objektum által támogatott tulajdonság.
- A tulajdonság beállításához használható *metódus* argumentum az objektum által támogatott metódus.

A tulajdonság egy OLE Automatizmus feladatban használni kívánt objektum megadására használható.

Az OLE Automatizmus feladatokban az **Object** tulajdonság által visszaadott objektum által támogatott tulajdonságok és metódusok használhatók. Hogy az adott objektum mely tulajdonságokat és metódusokat támogatja, az objektumot létrehozó alkalmazástól függ, és annak dokumentációjából tudható meg.

A tulajdonság a Szintaxis 2 szerint **CommonDialog** és **Grid** vezérlőelemeken használható, és egy felhasználói vezérlőelem azon tulajdonságait vagy metódusait adja vissza, amelyeknek ugyanazok a nevei, mint a Visual Basic által automatikusan a vezérlőelemre kiterjesztett tulajdonságoknak vagy metódusoknak. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *tulajdonság* argumentum a Visual Basic által szolgáltatott tulajdonság nevével azonos felhasználói objektum tulajdonság.
- A tulajdonság beállításához használható *metódus* argumentum a Visual Basic által szolgáltatott metódus nevével azonos felhasználói objektum metódus.
- A tulajdonság beállításához használható *érték* argumentum a tulajdonság vagy metódus konstansa vagy értéke.

A Visual Basic egy projektjében előforduló felhasználói vezérlőelem tulajdonságainak és metódusainak standard halmazát vagy annak egy részét bocsátja rendelkezésre. A felhasználói vezérlőelem számára mint a standard tulajdonsággal vagy metódussal azonos nevű tulajdonság vagy metódus adható meg. Ilyen esetben a Visual Basic automatikusan a standard tulajdonságot vagy metódust használná, azonban az **Object** tulajdonság lehetőséget ad arra, hogy ezt megkerülve a felhasználói vezérlőelemen a számára megadott tulajdonságot vagy metódust használjuk.

A Visual Basic a projektjében a következő tulajdonságokat és metódusokat automatikusan kiterjeszti a felhasználói vezérlőelemekre:

Tulajdonságok:

Align	**DragIcon**	**LinkMode**	**TabIndex**
Cancel	**DragMode**	**LinkItem**	**TabStop**
Container	**Enabled**	**LinkTimeout**	**Tag**
DataChanged	**Height**	**LinkTopic**	**Top**
DataField	**HelpContextID**	**Name**	**Visible**
DataSource	**Index**	**Negotiate**	**WhatsThisHelpID**
Default	**Left**	**Parent**	**Width**

Metódusok:

Drag	**LinkRequest**	**SetFocus**	**LinkExecute**
LinkSend	**ShowWhatIs**	**LinkPoke**	**Move**
ZOrder			

A felhasználói vezérlőelemen használt tulajdonság vagy metódus nem elvárt működése esetén elsőként ellenőrizni kell, hogy tulajdonságnak vagy metódusnak az előző lista valamelyik elemével ugyanazon neve van-e. Ha a nevek egyeznek, a felhasználói vezérlőelem dokumentációjában ellenőrizni kell, hogy működésmódja megegyezik-e a Visual Basic által szolgáltatott tulajdonságéval vagy metódusával. Ha a működésmód eltérő, az **Object** tulajdonság használatával lehet a felhasználói vezérlőelem kívánt képességéhez hozzáférni.

ObjectAcceptFormats

Szintaxis: *objektum*.**ObjectAcceptFormats**(*érték*)

A tulajdonság **OLE** befogadó vezérlőelemen érvényes, és az objektum által elfogadható formák listáját adja vissza. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a tömb elemét meghatározó numerikus kifejezés.

Az **ObjectAcceptFormats** egy 0 alapú karakterlánc-tömb, amely a **Format** tulajdonságban használható, a **Data** és **DataText** tulajdonságok használatával történő adatcsere során érvényes formák leírását tartalmazza. Az **ObjectAcceptFormats** tömb elemeinek számát az **ObjectAcceptFormatsCount** tulajdonság tartalmazza.

A használható formák listáját az objektum szolgáltatja. Ha az **OLE** befogadó vezérlőelem nem tartalmaz objektumot, az **ObjectAcceptFormats** tulajdonság használatának kísérlete hibát okoz.

ObjectAcceptFormatsCount

Szintaxis: *objektum*.**ObjectAcceptFormatsCount**

Ez a tulajdonság **OLE** befogadó vezérlőelemen érvényes, és az objektum által támogatott formák számát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonságot az **ObjectAcceptFormats** tulajdonságtömb elemszámának meghatározásához lehet használni.

ObjectGetFormats

Szintaxis: *objektum*. **ObjectGetFormats**(*érték*)

A tulajdonság **OLE** befogadó vezérlőelemen érvényes, és az objektum által szolgáltatott formák listáját adja vissza. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a tömb elemét meghatározó numerikus kifejezés.

Az **ObjectGetFormats** egy 0 alapú karakterlánc-tömb, amelynek elemeit a **Format** tulajdonság beállítására lehet használni a **Data** és **DataText** tulajdonságok használatával az objektumból történő adatátvétel során.

ObjectGetFormatsCount

Szintaxis: *objektum*.**ObjectGetFormatsCount**

Ez a tulajdonság **OLE** befogadó vezérlőelemen érvényes, és az objektum által szolgáltatott formák számát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonságot az **ObjectGetFormats** tulajdonságtömb elemszámának meghatározásához lehet használni.

ObjectVerbFlags

Szintaxis: *objektum*.**ObjectVerbFlags**(*érték*)

A tulajdonság **OLE** befogadó vezérlőelemen érvényes, és az **ObjectVerb** tömbjében az egyes parancsokhoz tartozó menüállapotra vonatkozó információkat adja vissza. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a tömb elemét meghatározó numerikus kifejezés.

A tulajdonság lehetséges értékei a következők:

Ha a szimbolikus konstans:	Értéke:	A menüállapot:
vbOLEFlagChecked	&H0008	A menüelem kiválasztott.
vbOLEFlagDisabled	&H0002	A menüelem letiltott, de nem szürke megjelenésű.
vbOLEFlagEnabled	&H0000	A menüelem kiválasztható.
vbOLEFlagGrayed	&H0001	A menüelem szürke megjelenítésű.
vbOLEFlagSeparator	&H0800	A menüelem egy elválasztósáv.

Ezek a konstansok a Visual Basic objektumtárban az Object Browser által adott listában találhatók.

Az **ObjectVerbs** tömb első eleme az alapértelmezett, a többi pedig a menün megjeleníthető parancs lesz. Az **ObjectVerbFlags** tömb az **ObjectVerbs** tömbben található minden egyes parancsához tartozó menüállapotról tartalmaz információt.

Egy objektum parancsait tartalmazó menü megjelenítésekor ezen tulajdonság értékeinek vizsgálatával a menüelem megjelenése határozható meg.

ObjectVerbs

Szintaxis: *objektum*.**ObjectVerbs**(*érték*)

A tulajdonság **OLE** befogadó vezérlőelemen érvényes, és az objektum által támo-

gatott menüparancsok listáját adja vissza. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a tömb elemét meghatározó numerikus kifejezés.

Az **ObjectVerbs** egy 0 alapú karakterlánc-tömb, amelynek elemei az **ObjectVerbsCount** tulajdonsággal együtt az objektum által támogatott parancsok kiolvasására használhatók. Ezek a parancsok az objektum **ObjectVerb** metódussal történő aktiválásakor végrehajtandó műveletek meghatározására használhatók. A parancsok listája objektumfüggő, és az aktuális körülményektől függ.

Minden egyes objektum saját parancshalmazt támogathat, azonban a következő táblázatbeli értékek az összes objektum által támogatott standard parancsokat jelölik:

Ha az *érték*:	Értelmezése:
0	Az objektum alapértelmezett művelete.
-1	Az objektum szerkesztésre aktivált. Ha az objektumot létrehozó alkalmazás támogatja a helybeni aktiválást, az objektum az **OLE** befogadó vezérlőelemen belül aktiválódik.
-2	Az objektumot egy különálló alkalmazási ablakban nyitja meg. Ha az objektumot létrehozó alkalmazás támogatja a helybeni aktiválást, az objektum saját ablakában aktiválódik.
-3	A beágyazott objektumoknál rejtetté teszi az objektumot létrehozó alkalmazást.
-4	Ha az objektum támogatja, az objektumot helybeni aktiválásra aktiválja, és láthatóvá teszi a felhasználói interfész-eszközöket. Ha az objektum nem támogatja a helybeni aktiválást, az objektum nem aktiválódik és hiba következik be.
-5	Ha a felhasználó a fókuszt egy **OLE** befogadó vezérlőelemre mozgatja, az objektum számára egy ablakot hoz létre, és az objektumot szerkesztésre készíti elő. Hiba következik be, ha az objektum nem támogatja az egyetlen kattintással történő aktiválást.
-6	Akkor használható, ha az objektum az alkalmazása által nem visszavonhatóan módosított rekordokat elhagyó szerkesztésre aktivált.

Ezek a parancsok nem szerepelnek az **ObjectVerbs** tömb listájában.

Az **ObjectVerbs** tömb első eleme az alapértelmezett **ObjectVerbs(0)** lesz., és — hacsak másként nincs előírva — ez a parancs aktiválja az objektumot, míg a többi parancs a menün jeleníthető meg. Ha az alapértelmezett parancsot a menün meg kívánjuk jeleníteni, ehhez az **ObjectVerbs** tömbben két bejegyzés tartozik.

Az objektumokat megjelenítő alkalmazások az Edit menüben jellemzően egy Object parancsot tartalmaznak, amelynek kiválasztásakor a menün az objektum paran-

csai jelennek meg. Futásidőben az **ObjectVerbs**, **ObjectVerbsCount** és **ObjectVerbFlags** tulajdonsággal lehet létrehozni ilyen menüt.

Az objektum által támogatott parancsok az objektum állapotától függően változhatnak. Az objektum által támogatott parancsok listáját a felhasználó számára történő felkínálás előtt a **FetchVerbs** metódussal kell frissíteni.

Ha az **AutoVerbMenu** tulajdonság **True**, az **ObjectVerbs** tömb parancsai a felhasználó által egy objektumon a JOBB egérgombbal történő kattintása hatására automatikusan egy előugró menüben jelennek meg.

ObjectVerbsCount

Szintaxis: *objektum*.**ObjectVerbsCount**

Ez a tulajdonság **OLE** befogadó vezérlőelemen érvényes, és az objektum által támogatott parancsok számát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonságot az **ObjectVerbs** tulajdonságtömb elemszámának meghatározásához lehet használni.

Az objektum által támogatott parancsok az objektum állapotától függően változhatnak. Az objektum által támogatott parancsok listáját a felhasználó számára történő felkínálás előtt a **FetchVerbs** metódussal kell frissíteni.

OLETimeout

Szintaxis: *objektum*.**OLETimeout** [= *érték*]

A tulajdonság **QueryDef** objektumon érvényes, és egy lekérdezés ODBC adatbázison történő futása során a várakozási idő értékének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a Microsoft Jet adatbázis-szerkesztőnek a várakozási idő túllépés hiba kiadása előtt szükséges idő másodpercekben adott, **Integer** típusú értéke.

Egy ODBC — például SQL Server — adatbázis használata során a hálózati forgalom vagy a kiszolgáló leterheltsége miatt késlekedés léphet fel. A határozatlan ideig tartó várakozás helyett megadhatjuk a Microsoft Jet hibajelzés előtti várakozási idejét.

A Microsoft Jet számára a hibajelzés kiadása előtti alapértelmezett 60 másodperces várakozási idő helyett más időtartam is megadható. A **Database** objektum **QueryTimeout** tulajdonságának beállítása felülírja az alapértelmezett értéket.

OLEDropAllowed

Szintaxis: *objektum*.**OLEDropAllowed** [= *érték*]

A tulajdonság **OLE** befogadó vezérlőelemen érvényes, és a vezérlőelemnek az OLE "elhúzás-és-leejtés" műveletében leejtési célként történő használatát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a leejtési célként történő használatot meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True Egy csatolható vagy beágyazható objektum elhúzásakor egy leejtés ikon jelenik meg, amikor az egérmutató egy **OLE** befogadó vezérlőelem felett halad el. Az objektumnak egy **OLE** vezérlőelembe ejtése ugyanolyan hatású, mintha azt a Clipboardról a **Paste** metódussal illesztenénk be.

False Nem jelenik meg leejtés ikon, amikor egy csatolható vagy beágyazható objektum elhúzásakor az egérmutató egy **OLE** befogadó vezérlőelem felett halad el. Az objektum **OLE** vezérlőelembe ejtésének nincs hatása a vezérlőelemre. (Alapértelmezett)

Az **OLEDropAllowed** tulajdonság **True** értéke esetén a megjelenő egérmutató alakját a **MousePointer** tulajdonság határozza meg. Ha annak értéke az alapértelmezett 0, a Visual Basic a művelet során a standard "elhúzás-és-leejtés" ikont jeleníti meg.

A csatolható és beágyazható objektum mozgatásakor vagy másolásakor az **OLETypeAllowed** tulajdonságnak 1 (**vbOLEEmbedded**) vagy 2 (**vbOLEEither**), az objektum csatolásához pedig 0 (**vbOLELinked**) vagy 2 (**vbOLEEither**) beállításúnak kell lennie. Az **OLEDropAllowed** tulajdonság **True** értéke esetén egy objektum leejtése a **Class**, **SourceDoc** és **SourceItem** tulajdonságok beállítására az **OLE** befogadó vezérlőelem **Paste** metódusának használatával azonos hatású.

Ha az **OLEDropAllowed** tulajdonság **True**, az **OLE** befogadó vezérlőelem egy objektum elhúzásakor nem fogadja a **DragDrop** és **DragOver** eseményeket, és **DragMode** tulajdonságnak nincs hatása az **OLE** befogadó vezérlőelem "elhúzás-és-leejtés" viselkedésére.

OLERequestPendingMsgText

Szintaxis: *objektum*.**OLERequestPendingMsgText** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **App** objektumon érvényes, és egy OLE Automatizmus kérésre várakozás során egérről vagy billentyűzetről történő bevitel esetén megjelenő foglaltsági üzenet szövegének kiolvasására vagy beállí-

tására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az OLE kérésre várakozás során a helyettesítő üzenetpanelen megjelenő szöveget tartalmazó karakteres kifejezés. Ha a kifejezés üres ("") karakterlánc, az alapértelmezett Server Busy párbeszédpanel jelenik meg.

A Visual Basic egy alapértelmezett Server Busy párbeszédpanelt jelenít meg, ha egy OLE Automatizmus kérésre várakozás alatt egérről vagy billentyűzetről történő bevitel történik. Ez a párbeszédpanel szöveget és egy látható OLE kiszolgálóval (például Microsoft Excellel) történő használatra szánt Switch To gombot tartalmaz. A következő körülmények között az alapértelmezett párbeszédpanel nem mindenben felelhet meg kívánalmainknak:

- Ha a programunk a felhasználói felülettel nem rendelkező OLE kiszolgáló által szolgáltatott objektum egy metódusát hívja meg. Például a Visual Basic Professional változat használatával létrehozott OLE kiszolgálók bármilyen látható űrlap nélkül, a háttérben futhatnak.
- Ha a hívott OLE kiszolgáló a Visual Basic Enterprise változatának Remote Automation jellemzőjével volt létrehozva, és a felhasználótól távoli gépen fut.
- Ha a programunk a **GetObject( )** függvény használatával egy Microsoft Excel munkafüzetet töltött be, mivel a munkafüzet nem lesz látható, ha a felhasználó az Excelre kapcsol át. Lényegében az Excel maga sem látható, amely esetben a Switch To gomb hatástalan.

Az ilyen körülmények között az alapértelmezett szöveg és a Switch To gomb nem megfelelő és a programunk használójának nem ad egyértelmű tájékoztatást.

Az **OLERequestPendingMsgText** tulajdonság lehetővé teszi az alapértelmezett Server Busy párbeszédpanelnek egy másik panellel történő helyettesítését. A tulajdonságba saját üzenetet beállítva az alapértelmezett Server Busy párbeszédpanel helyett az *érték* argumentum karakterláncát és egy OK gombot tartalmazó egyszerű párbeszédpanel jelenik meg.

Ha egy OLE Automatizmus kérést az OLE kiszolgáló már elfogadta, a kérés már nem törölhető.

Amikor tudjuk, hogy egy OLE Automatizmus kérés néhány másodpercnél több időt igényelhet és rejtett vagy távoli OLE kiszolgálót használunk, egy helyettesítő üzenetet állíthatunk be. A távoli OLE kiszolgáló esetén a helyettesítő üzenet használata minden kéréshez javasolt. A hálózati forgalom esetenként még a nagyon rövid OLE kérésnél is több másodperces késleltetést okozhat.

OLERequestPendingMsgTitle

Szintaxis: *objektum*.**OLERequestPendingMsgTitle** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **App** objektumon érvényes, és egy OLE Automatizmus kérésre várakozás során egérről vagy billentyűzetről történő bevitel esetén megjelenő foglaltsági üzenet címsorszövegének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az OLE kérésre várakozás során a helyettesítő üzenetpanel címsorának szövegét tartalmazó karakteres kifejezés. Ha a kifejezés üres ("") karakterlánc, az **OLERequestPendingMsgTitle** tulajdonság figyelmen kívül marad.

Ha az **OLERequestPendingMsgText** tulajdonság beállított, a helyettesítő foglaltsági üzenet címsora az **OLERequestPendingMsgTitle** értékével adható meg. Az alapértelmezett — és javasolt — cím az **App** objektum **Title** tulajdonságának értéke.

OLERequestPendingTimeout

Szintaxis: *objektum*.**OLERequestPendingTimeout** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **App** objektumon érvényes, és egy OLE Automatizmus kérésre várakozás során egérről vagy billentyűzetről történő bevitel esetén a foglaltsági üzenet megjelenéséig előírt várakozási idő értékének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a foglaltsági üzenet megjelenéséig előírt várakozási idő milliszekundumban adott, **Long** típusú értéke. Az alapértelmezett várakozási idő 5000 milliszekundum, azaz 5 másodperc.

A várakozási idő az **OLE** befogadó vezérlőelem vagy a Toolbox parancs használatával csatolt vagy beágyazott dokumentumokra is hatással van. Ha csatolt vagy beágyazott dokumentumot használunk és az **OLERequestPendingTimeout** értékét egy OLE Automatizmus kérést megelőzően módosítjuk, célszerű utána az értéket újra beállítani.

OLEServerBusyMsgText

Szintaxis: *objektum*.**OLEServerBusyMsgText** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **App** objektumon érvé-

nyes, és egy OLE Automatizmus kérés OLE kiszolgáló általi elutasítása esetén megjelenő helyettesítő üzenet szövegének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az OLE kiszolgáló foglaltsági körülményei esetén a helyettesítő üzenetpanelen megjelenő szöveget tartalmazó karakteres kifejezés. Ha a kifejezés üres ("") karakterlánc, az alapértelmezett Server Busy párbeszédpanel jelenik meg.

A Visual Basic az **OLERequestPendingTimeout** tulajdonságban adott időtartamig folytonosan ismétlődően próbálkozik egy OLE Automatizmus kéréssel. Ha az OLE kiszolgáló az adott időn belül nem fogadja a kérést, egy alapértelmezett Server Busy párbeszédpanel jelenik meg. Ez a párbeszédpanel szöveget és egy látható OLE kiszolgálóval (például Microsoft Excellel) történő használatra szánt Switch To gombot tartalmaz. A következő körülmények között az alapértelmezett párbeszédpanel nem mindenben felelhet meg kívánalmainknak:

- Ha a programunk a felhasználói felülettel nem rendelkező OLE kiszolgáló által szolgáltatott objektum egy metódusát hívja meg. Például a Visual Basic Professional változat használatával létrehozott OLE kiszolgálók bármilyen látható űrlap nélkül, a háttérben futhatnak.
- Ha a hívott OLE kiszolgáló a Visual Basic Enterprise változatának Remote Automation jellemzőjével volt létrehozva, és a felhasználótól távoli gépen fut.
- Ha a programunk a **GetObject( )** függvény használatával egy Microsoft Excel munkafüzetet töltött be, mivel a munkafüzet nem lesz látható, ha a felhasználó az Excelre kapcsol át. Lényegében az Excel maga sem látható, amely esetben a Switch To gomb hatástalan.

Az ilyen körülmények között az alapértelmezett szöveg és a Switch To gomb nem megfelelő és a programunk használójának nem ad egyértelmű tájékoztatást.

Az **OLEServerBusyMsgText** tulajdonság lehetővé teszi az alapértelmezett Server Busy párbeszédpanelnek egy másik panellel történő helyettesítését. A tulajdonságba saját üzenetet beállítva az alapértelmezett Server Busy párbeszédpanel helyett az *érték* argumentum karakterláncát, valamint egy OK és egy Cancel gombot tartalmazó egyszerű párbeszédpanel jelenik meg.

Ha a felhasználó akár az alapértelmezett Server Busy párbeszédpanelen, akár a helyettesítő párbeszédpanelen lenyomja a Cancel gombot, az OLE Automatizmus kérést kiadó eljárásban a -2147418111 (&H800100001) OLE hiba következik be.

Amikor tudjuk, hogy egy OLE Automatizmus kérés néhány másodpercnél több időt igényelhet és rejtett vagy távoli OLE kiszolgálót használunk, egy helyettesítő üzenetet állíthatunk be. A távoli OLE kiszolgáló esetén a helyettesítő üzenet használata minden kéréshez javasolt. A hálózati forgalom esetenként még a nagyon rövid OLE kérésnél is

több másodperces késleltetést okozhat.

OLEServerBusyMsgTitle

Szintaxis: *objektum*.**OLEServerBusyMsgTitle** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **App** objektumon érvényes, és egy OLE Automatizmus kérés OLE kiszolgáló általi elutasítása esetén megjelenő helyettesítő foglaltsági üzenet címsorának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az OLE kérésre várakozás során a helyettesítő üzenetpanel címsorának szövegét tartalmazó karakteres kifejezés. Ha a kifejezés üres ("") karakterlánc, az **OLEServerBusyMsgTitle** tulajdonság figyelmen kívül marad.

Ha az **OLEServerBusyMsgText** tulajdonság beállított, a helyettesítő foglaltsági üzenet címsora az **OLEServerBusyMsgTitle** értékével adható meg. Az alapértelmezett — és javasolt — cím az **App** objektum **Title** tulajdonságának értéke.

OLEServerBusyRaiseError

Szintaxis: *objektum*.**OLEServerBusyRaiseError** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **App** objektumon érvényes, és egy OLE Automatizmus kérés OLE kiszolgáló általi elutasítása esetén a foglaltsági üzenet megjelenítése helyett egy hiba előidézését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a hiba előidézését meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True Az **OLEServerBusyTimeout** tulajdonságban adott idő letelte után hiba következik be.

False Az **OLEServerBusyMsgText** tulajdonság beállításától függően vagy az alapértelmezett Server Busy párbeszédpanel, vagy a helyettesítő foglaltsági üzenet jelenik meg.

Egy OLE Automatizmus kérés OLE kiszolgáló általi elutasítása esetén bekövetkező hibánál a vezérlés visszakerül a programunkra, amely az alapértelmezett Server Busy

párbeszédpanel, vagy a helyettesítő foglaltsági üzenet helyett saját, testre szabott párbeszédpanel megjelenítését teszi lehetővé.

A bekövetkező hiba az OLE Automatizmus kérést kiadó eljárásban -2147418111 (&H800100001) lesz.

OLEServerBusyTimeout

Szintaxis: *objektum*.**OLEServerBusyTimeout** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **App** objektumon érvényes, és egy OLE Automatizmus kérés ismétlődő kísérletének a foglaltsági üzenet megjelenéséig tartó idejét meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy OLE Automatizmus kérés ismétlődő kísérletére előírt idő milliszekundumban adott, **Long** típusú értéke. Az alapértelmezett idő 10000 milliszekundum, azaz 10 másodperc.

Az adott idő az **OLE** befogadó vezérlőelem vagy a Toolbox parancs használatával csatolt vagy beágyazott dokumentumokra is hatással van. Ha csatolt vagy beágyazott dokumentumot használunk és az **OLEServerBusyTimeout** értékét egy OLE Automatizmus kérést megelőzően módosítjuk, célszerű utána az értéket újra beállítani.

OLEType

Szintaxis: *objektum*.**OLEType**

Ez tulajdonság **OLE** befogadó vezérlőelemen érvényes, és az objektum vezérlőelemen belüli állapotát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság értékei a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	Az állapot:
vbOLENone	3	*None* — Az **OLE** befogadó vezérlőelem nem tartalmaz objektumot.
vbOLELinked	0	*Linked* — Az **OLE** befogadó vezérlőelem csatolt objektumot tartalmaz. Az objektum minden adatát az őt létrehozó alkalmazás szervezi. Amikor az objektum a **SaveToFile** metódussal mentett, csak a **SourceDoc**, **SourceItem** és hasonló csatolási információk kerülnek a Visual Basic alkalmazás által meghatározott állományba.

vbOLEEmbedded	1	*Embedded* — Az **OLE** befogadó vezérlőelem beágyazott objektumot tartalmaz. Az OLE objektum minden adatát a Visual Basic alkalmazás szervezi. Amikor az objektum a **SaveToFile** metódussal mentett, minden, az objektummal kapcsolatos információ az adott fájlba kerül.

A tulajdonságot annak meghatározására lehet használni, hogy egy **OLE** befogadó vezérlőelem tartalmaz-e objektumot és hogy az milyen típusú.

Az objektumot létrehozó alkalmazás futását az **AppIsRunning** tulajdonság értéke jelzi. A létrehozható objektumok típusait az **OLETypeAllowed** tulajdonság tartalmazza.

OLETypeAllowed

Szintaxis: *objektum*.**OLETypeAllowed** [= *érték*]

A tulajdonság **OLE** befogadó vezérlőelemen érvényes, és a vezérlőelemben megengedett objektumtípusok kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az objektumtípust meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	Értéke:	Az objektumtípus:
vbOLELinked	0	*Linked* — Az **OLE** befogadó vezérlőelem csak egy csatolt objektumot tartalmazhat.
vbOLEEmbedded	1	*Embedded* — Az **OLE** befogadó vezérlőelem csak egy beágyazott objektumot tartalmazhat.
vbOLEEither	2	*Either* — Az **OLE** befogadó vezérlőelem egy beágyazott vagy egy csatolt objektumot tartalmazhat. (Alapértelmezett.)

A tulajdonság a felhasználó által létrehozható objektumokat határozza meg:

- Az Insert Object párbeszédpanel használatakor a párbeszédpanel megjelenítéséhez az **InsertObjDlg** metódust kell használni.
- A Paste Special párbeszédpanel használatakor a párbeszédpanel megjelenítéséhez a **PasteSpecialDlg** metódust kell használni.
- Egy objektumnak a rendszer vágólapjáról történő beillesztéséhez a **Paste** metódust kell használni.

Az objektumtípus (csatolt, beágyazott vagy egyik sem) meghatározására az

OLEType tulajdonság használható.

Options

Szintaxis: *objektum*.**Options** [= *érték*]

A tulajdonság **Data** vezérlőelemen érvényes, és a vezérlőelem **Recordset** tulajdonságában adott **Recordset** objektum egy vagy több jellemzőjét meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Recordset** egy jellemzőjét meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	Értéke:	A keresési mód:
dbDenyWrite	1	Többfelhasználós környezetben a többi felhasználó nem módosíthatja a **Recordset** rekordjait.
dbDenyRead	2	Többfelhasználós környezetben a többi felhasználó nem olvashatja a tábla típusú **Recordset** rekordjait.
dbReadOnly	4	A **Recordset** rekordjai nem módosíthatók.
dbAppendOnly	8	A **Recordset** objektumhoz új rekordok csatolhatók, azonban a meglévők nem olvashatók.
dbInconsistent	16	A **Recordset** minden mezője frissíthető, még akkor is, ha azok megsértik a kapcsolódási feltételeket.
dbConsistent	32	Csak azok a mezők frissíthetők, amelyek nem sértik a kapcsolódási feltételeket. (Alapértelmezett.)
dbSQLPassThrough	64	Ha egy **Data** vezérlőelemet SQL utasítással együtt használunk a **RecordSource** tulajdonságban, az utasítást feldolgozásra egy ODBC (például SQL Server vagy Oracle) adatbázisnak kell küldeni.
dbForwardOnly	256	A **Recordset** előre görgethető, az egyetlen érvényes mozgatási metódusa a **MoveNext**. Ez a beállítás nem használható a **Data** vezérlőelemmel kezelt **Recordset** objektumokon.
dbSeeChanges	512	Kezelhető hibát okoz, ha másik felhasználó módosítja az általunk szerkesztett adatot.

Ezek a konstansok a Visual Basic objektumtárban az Object Browser által adott

listában találhatók.

Az **Options** tulajdonság futásidejű módosítását követően a módosítást a **Refresh** metódussal kell érvényessé tenni.

Egyidőben a tulajdonság több értéke a szokásos módon, a szimbolikus konstansok vagy értékek összegeként állítható be. Annak meghatározására, hogy a tulajdonság tartalmaz-e egy adott értéket, az **And** operátort használhatjuk, például az `If Adat1.Options And dbAppendOnly Then ....` alakban.

Amennyiben a **dbInconsistent** és **dbConsistent** érték is beállított, a kettő közül az alapértelmezett **dbConsistent** lesz érvényes.

A **dbSQLPassThrough** beállítás csak eredményhalmaz vagy pillanatfelvétel típusú **Recordset** objektumok létrehozásakor használható és csak az előző verziókkal való kompatibilitás miatt támogatott. Helyette az előzetesen létrehozott SQL átadó **QueryDef** objektum használható és a **Data** vezérlőelem **Recordset** tulajdonságát a **QueryDef** objektummal létrehozott **Recordset** objektumra kell beállítani.

OrdinalPosition

Szintaxis: *objektum*.**OrdinalPosition** [= *érték*]

A tulajdonság **Field** objektumon érvényes, és a **Field** objektumnak a **Fields** objektumgyűjteményén belüli relatív pozíciójának kiolvasására vagy beállítására használható. Objektumgyűjteményhez még nem tartozó objektumon a tulajdonság írható és olvasható is. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a mezők számszerinti sorrendjét meghatározó **Integer** típusú érték.

Az **OrdinalPosition** tulajdonság használhatóságát a **Field** objektum **Fields** halmazát tartalmazó objektum a következők szerint határozza meg:

Ha az objektum:	A tulajdonság használata:
Index	Nem támogatott.
QueryDef	Csak olvasható.
Recordset	Csak olvasható.
Relation	Nem támogatott.
TableDef	Írható és olvasható.

Általában egy halmazhoz csatolt objektum számszerinti sorrendje az objektumhoz csatolás sorrendjétől függ: az elsőként csatolt objektum pozíciója 0, a másodiké 1, és így tovább, míg az utolsó objektum pozíciója a **Count**-1 érték lesz. Ennél nagyobb **OrdinalPosition** értéket is megadhatunk, és ez esetben a visszaadott mező meghatározá-

sa a legnagyobb számhoz mért relatív pozíció alapján történik.

Az **OrdinalPosition** tulajdonság használatával ez a természetes sorrend módosítható, ezáltal meghatározható egy alkalmazásban használt táblák, lekérdezések és rekordhalmazok mezőinek sorrendje. A tulajdonság beállításával a rekordhalmazból visszaadott rekordok sorrendje bármikor beállítható. A Microsoft Jet bármely pozitív értéket elfogad az **OrdinalPosition** tulajdonság értékeként, a negatív számok azonban kezelhető hibát okoznak.

Ugyanazon halmazban két vagy több **Field** objektumnak is lehet ugyanazon **OrdinalPosition** értéke, ekkor az objektumok rendezése betűrend szerint történik.

Az **OrdinalPosition** beállítás módosítása után a tulajdonság az aktuális beállítást adja vissza, azonban a mezők átrendezése a **Fields** halmaz újbóli feltöltéséig nem történik meg.

A **Fields** halmazban egy objektum **OrdinalPosition** tulajdonságának módosítását a **Refresh** metódussal kell érvényessé tenni.

Orientation

Szintaxis: *objektum*.**Orientation** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Printer** objektumon érvényes, és a dokumentum álló vagy fekvő formában történő nyomtatását meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a lap nyomtatáskor érvényes irányítottságát (álló vagy fekvő) meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	Értéke:	A nyomtatott lap orientációja:
vbPRORPortrait	1	*Portrait* — Az oldal teteje a lap keskenyebb oldalára esik.
vbPRORLandscape	2	*Landscape* — Az oldal teteje a lap szélesebb oldalára esik.

Ezek a konstansok a Visual Basic objektumtárban az Object Browser által adott listában találhatók.

A **Printer** objektum tulajdonságainak hatása a nyomtató gyártója által adott vezérlőprogramtól függ: egyes tulajdonságok hatástalanok lehetnek vagy több tulajdonság azonos hatást eredményezhet. Erre vonatkozóan a nyomtató leírása tartalmazhat további információt.

Owner

Szintaxis: *objektum*.**Owner** [= *érték*]

Ez a tulajdonság **Container** és **Document** objektumon érvényes, és az objektum tulajdonosát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum vagy a **Users** halmaz egy **User** objektumának, vagy a **Groups** halmaz egy **Group** objektumának tulajdonosát meghatározó **String** típusú kifejezés.

Egy objektumhoz történő hozzáférés szempontjából az objektum tulajdonosának a többi felhasználóval szemben bizonyos elsőbbsége van. Az **Owner** tulajdonság beállítása bármikor, bármely — megfelelő jogosultsággal rendelkező — egyedi (egy **User** objektummal jelölt) vagy csoportos (egy **Group** objektummal jelölt) felhasználói számláról lehetséges.

Page

Szintaxis: *objektum*.**Page**

Ez tulajdonság **Printer** objektumon érvényes, és az aktuális oldal számát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonság az alkalmazás indítása vagy a **Printer** objektumon utoljára használt **EndDoc** utasítás végrehajtása óta nyomtatott oldalakat egytől kezdődően számozza, és az oldalszám eggyel nő:

- Minden **NewPage** utasítást követően.
- A **Print** metódus használata után is akkor is eggyel nő az oldalszám, ha a nyomtatandó szöveg nem illeszkedett az aktuális oldalhoz.

A grafikus metódusok oldalhoz nem illeszkedő kimenete nem okoz oldalszámváltozást, a kimenet egyszerűen csak a nyomtatható területnek megfelelő nagyságra vágott lesz.

PaperBin

Szintaxis: *objektum*.**PaperBin** [= *érték*]

Ez a tulajdonság **Printer** objektumon érvényes, és a nyomtató alapértelmezett papírtárolóját meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az alapértelmezett papíradagolási módot meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	Értéke:	A papírellátás módja:
vbPRBNUpper	1	A papíradagolás a felső tárolóból történik.
vbPRBNLower	2	A papíradagolás az alsó tárolóból történik.
vbPRBNMiddle	3	A papíradagolás a középső tárolóból történik.
vbPRBNManual	4	A papíradagolás laponként, kézzel történik.
vbPRBNEnvelope	5	A boríték adagolása a borítéktárolóból történik.
vbPRBNEnvManual	6	A boríték adagolása a borítéktárolóból egyenként, kézzel történik.
vbPRBNAuto	7	A papíradagolás az aktuális papírtárolóból történik. (Alapértelmezett.)
vbPRBNTractor	8	A papíradagolás traktor típusú továbbítóegységgel történik.
vbPRBNSmallFmt	9	A papíradagolás a kis papírtovábbító egységből történik.
vbPRBNLargeFmt	10	A papíradagolás a nagy papírtároló egységből történik.
vbPRBNLargeCapacity	11	A papíradagolás a nagy kapacitású papírtároló egységből történik.
vbPRBNCasette	14	A papíradagolás a kiegészítő kazettás papírtároló egységből történik.

Ezek a konstansok a Visual Basic objektumtárban az Object Browser által adott listában találhatók.

Természetesen a beállítható papíradagolási módok nem mindegyike használható minden nyomtatónál, az érvényes beállítások a nyomtató dokumentációjából tudhatók meg.

A **Printer** objektum tulajdonságainak hatása a nyomtató gyártója által adott vezérlőprogramtól függ: egyes tulajdonságok hatástalanok lehetnek vagy több tulajdonság azonos hatást eredményezhet. Erre vonatkozóan a nyomtató leírása tartalmazhat további információt.

PaperSize

Szintaxis: *objektum*.**PaperSize** [= *érték*]

Ez a tulajdonság **Printer** objektumon érvényes, és az aktuális nyomtató papírmére-

tét meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a papírméretet meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	Értéke:	A papírméret:
vbPRPSLetter	1	Letter (8 1/2 x 11 inch)
vbPRPSLetterSmall	2	Letter Small (8 1/2 x 11 inch)
vbPRPSTabloid	3	Tabloid (11 x 17 inch)
vbPRPSLedger	4	Ledger (17 x 11 inch)
vbPRPSLegal	5	Legal (8 1/2 x 14 inch)
vbPRPSStatement	6	Statement (5 1/2 x 8 1/2 inch)
vbPRPSExecutive	7	Executive (7 1/2 x 11 1/2 inch)
vbPRPSA3	8	A 3 (297 x 420 mm)
vbPRPSA4	9	A 4 (210 x 297 mm)
vbPRPSA4Small	10	A 4 Small (210 x 297 mm)
vbPRPSA5	11	A 5 (148 x 210 mm)
vbPRPSB4	12	B 4 (250 x 354 mm)
vbPRPSB5	13	B 5 (182 x 257 mm)
vbPRPSFolio	14	Folio (8 1/2 x 13 inch)
vbPRPSQuatro	15	Quatro (215 x 275 mm)
vbPRPS10x14	16	10 x 14 inch
vbPRPS11x17	17	11 x 17 inch
vbPRPSNote	18	Note (8 1/2 x 11 inch)
vbPRPSEnv9	19	Envelope #9 (3 7/8 x 8 7/8 inch)
vbPRPSEnv10	20	Envelope #10 (4 1/8 x 9 1/2 inch)
vbPRPSEnv11	21	Envelope #11 (4 1/2 x 10 3/8 inch)
vbPRPSEnv12	22	Envelope #12 (4 1/2 x 11inch)
vbPRPSEnv14	23	Envelope #14 (5 x 11 1/2 inch)
vbPRPSCSheet	24	C méretű lap
vbPRPSDSheet	25	D méretű lap
vbPRPSESheet	26	E méretű lap
vbPRPSEnvDL	27	Envelope DL (210 x 220 mm)
vbPRPSEnvC3	28	Envelope C3 (324 x 458 mm)
vbPRPSEnvC4	29	Envelope C4 (229 x 324 mm)
vbPRPSEnvC5	30	Envelope C5 (162 x 229 mm)
vbPRPSEnvC6	31	Envelope C6 (114 x 162 mm)
vbPRPSEnvC65	32	Envelope C65 (114 x 229 mm)
vbPRPSEnvB4	33	Envelope B4 (250 x 353 mm)

vbPRPSEnvB5	34	Envelope B5 (176 x 250 mm)
vbPRPSEnvB6	35	Envelope B6 (176 x 125 mm)
vbPRPSEnvItaly	36	Envelope (210 x 220 mm)
vbPRPSEnvMonarch	37	Envelope Monarch (3 7/8 x 7 1/2 inch)
vbPRPSEnvPersonal	38	Envelope (3 5/8 x 6 1/2 inch)
vbPRPSFanfoldUS	39	US Standard Fanfold (14 7/8 x 11 inch)
vbPRPSFanfoldStdGermany	40	German Standard Fanfold (8 1/2 x 12 inch)
vbPRPSFanfoldLglGermany	41	German Legal Fanfold (8 1/2 x 13 inch)
vbPRPSUser	256	Felhasználó által definiált

Ezek a konstansok a Visual Basic objektumtárban az Object Browser által adott listában találhatók.

A nyomtató **Height** vagy **Width** tulajdonsága a **PaperSize** tulajdonságot automatikusan **vbPRPSUser** értékre állítja.

A **Printer** objektum tulajdonságainak hatása a nyomtató gyártója által adott vezérlőprogramtól függ: egyes tulajdonságok hatástalanok lehetnek vagy több tulajdonság azonos hatást eredményezhet. Erre vonatkozóan a nyomtató leírása tartalmazhat további információt.

Parent

Szintaxis: *objektum*.**Parent**

Ez a tulajdonság **Application**, **Component**, **FileControl**, **FormTemplate**, **MenuLine**, **ProjectTemplate** és **Property** objektumokon, **ControlTemplates**, **MenuItems**, **Properties**, **SelectedComponents** és **SelectedControls** objektumgyűjteményeken, valamint **CheckBox**, **ComboBox**, **CommandButton**, **CommonDialog**, **Data**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **Line**, **ListBox**, **Menu**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox**, **Timer** és **VScrollBar** vezérlőelemeken érvényes, és egy másik vezérlőelemet, objektumot vagy objektumgyűjteményt tartalmazó űrlapot, objektumot vagy objektumgyűjteményt ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Patent** tulajdonságot egy objektum szülőjének tulajdonságaihoz, metódusaihoz vagy vezérlőelemeihez való hozzáférésre használható. A tulajdonság különösen olyan alkalmazásokban hasznos, ahol vezérlőelemeket argumentumként kell átadni.

A **Parent** és az **MDIChild** tulajdonság között nincs kapcsolat, azonban egy MDI-űrlap és bármely **True** beállítású **MDIChild** tulajdonságú űrlap között létezik szülő-leszármazó kapcsolat.

Password

Szintaxis:

Ez a **User** és **Workspace** objektumon érvényes tulajdonság a Visual Basic 4.0-s verzióban nem támogatott. A jelszó beállítása a **Workspace** objektumon a **CreateWorkspace** metódussal, míg a **Password** tulajdonság módosítása egy létező **User** objektumon a **NewPassword** metódussal lehetséges.

PasswordChar

Szintaxis: *objektum*.**PasswordChar** [= *érték*]

Ez a tulajdonság **TextBox** vezérlőelemen érvényes, és a felhasználó által beírt karaktereknek vagy a helyfoglalóknak a **TextBox** vezérlőelemben történő megjelenését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a helyfoglaló karaktert meghatározó karakteres kifejezés.

Ezzel a tulajdonsággal egy párbeszédpanelen jelszómező hozható létre. Helyfoglalóként bármely karaktert megadhatunk, azonban a Windows alapú alkalmazások a `Chr$(42)` karakterkódú csillag (*) karaktert használják.

A **PasswordChar** nem befolyásolja a **Text** tulajdonságot, az kizárólag a felhasználó által beírt vagy a program által beállított karaktereket tartalmazza. Az aktuális szöveg megjelenítéséhez a **PasswordChar** tulajdonságba az alapértelmezett üres karaktert ("") kell beállítani.

Noha a tulajdonságba bármely karakterlánc elhelyezhető, annak csak az első karaktere számít, a többi figyelmen kívül marad.

PasteOK

Szintaxis: *objektum*.**PasteOK**

Ez az **OLE** befogadó vezérlőelemen érvényes tulajdonság a Clipboard tartalmának az **OLE** befogadó vezérlőelembe történő beilleszthetőségét meghatározó értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Ha a tulajdonság értéke **True**, a Clipboard tartalma az **OLE** befogadó vezérlőelembe illeszthető.

Az OLE befogadó vezérlőelembe illeszthető objektumok (csatolt vagy beágyazott) típusait az **OLETypeAllowed** tulajdonság, egy objektum sikeres beillesztésével létrejött

objektum típusát pedig az **OLEType** tulajdonság beállításának vizsgálatával határozhatjuk meg.

A tulajdonság akkor használható, ha az alkalmazásunk támogatja az Edit menü Paste parancsát: az a **PasteOK** tulajdonság **False** értékével letiltható, **True** értékével pedig engedélyezhető. A menüparancsokat az **Enabled** tulajdonságuk beállításával lehet letiltani (**False**) vagy engedélyezni (**True**).

Egy objektum OLE befogadó vezérlőelembe illesztése a **Paste** metódussal történik.

A felhasználó számára nagyobb szabadságot biztosít, ha az Edit menü Paste Special parancsának kiválasztásakor az **OLETypeAllowed** = 2 érték beállítása után a **PasteSpecialDlg** metódus használatával megjelenítjük a Paste Special párbeszédpanelt, amelyen elvégezhetők a beillesztéssel kapcsolatos beállítások.

Path

Szintaxis: *objektum*.**Path** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **App** objektumon, valamint **DirListBox** és **FileListBox** vezérlőelemen érvényes, és az aktuális elérési út kiolvasására vagy beállítására használható. A tulajdonság az **App** objektumon futásidőben csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az elérési utat tartalmazó **String** típusú kifejezés.

A **Path** tulajdonság értéke az elérési utat tartalmazó karakterlánc, például C:\WINDOWS\SYSTEM. A **DirListBox** és **FileListBox** esetén az alapértelmezett elérési út a vezérlőelem futásidőben történt létrehozásakor érvényes út, az **App** objektumon pedig a programfejlesztési környezetben történő futtatáskor a projekt .VBP fájl, az alkalmazás futtatható fájlkénti használatakor pedig a .EXE fájl elérési útja lesz.

A tulajdonságot olyan alkalmazások építéséhez lehet használni, amely fájlok közötti tallózást és fájlmanipulációs lehetőségeket tartalmaz. A **Path** beállítás hasonló, mint a DOS operációs rendszerből ismert CHDIR parancs: relatív (meghajtónevet nem kötelezően használó) elérési út is használható, a *meghajtó*: alakú elérési út pedig az adott meghajtó aktuális könyvtárát adja vissza.

A **Path** hálózati elérési úthoz a \\ *kiszolgáló* \ *megosztási_név* \ *elérési_út* alakban használható. Ez a szintaxis a **Drive** tulajdonság értékébe üres ("") karakterláncot állít be.

A **Path** értékének módosítása a **DirListBox** vezérlőelem esetén **Change**, a **FileListBox** vezérlőelem esetén pedig **PathChange** eseményt is előidéz.

A **DirListBox** vezérlőelem esetén a **Path** tulajdonság visszaadott értéke eltér a kiválasztást tükröző `List(ListIndex)` által adott eredménytől.

Pattern

Szintaxis: *objektum*.**Pattern** [= *érték1* [; *érték2*]...]

A tulajdonság **FileListBox** vezérlőelemen érvényes, és a vezérlőelemen futásidőben megjelenített fájlneveket meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a fájlt meghatározó karakteres kifejezés, például "*.FRM", vagy az alapértelmezett, minden fájlnevet visszaadó "*.*" alakú lehet. A * és ? helyettesítő (wildcard) karakterek mellett egynél több, pontosvesszővel elválasztott névminta is megadható, például "*.EXE; "*.BAT" alakban.

A **Pattern** tulajdonság az olyan alkalmazásokban játszik lényeges szerepet, amelyek fájlok közötti tallózást és fájlmanipulációs lehetőségeket tartalmaznak. A tulajdonság a **Drive**, **FileName** és **Path** tulajdonságokkal együtt a felhasználó számára lehetőséget biztosít fájlok és fájlcsoportok keresésére és kezelésére.

A **Pattern** tulajdonság módosítása **PatternChange** eseményt idéz elő.

PercentPosition

Szintaxis: *objektum*.**PercentPosition** [= *érték*]

A tulajdonság **Recordset** objektumon érvényes, és az aktuális rekordnak az objektum rekordjain belüli közelítő helyzetét jelző vagy módosító érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy 0.0 és 100.00 közötti, **Single** típusú érték.

A **PercentPosition** tulajdonsággal ellenőrizhető vagy módosítható az aktuális rekordnak egy **Recordset** objektumon belüli közelítő pozíciója. A eredményhalmaz vagy pillanatfelvétel típusú **Recordset** esetén a **PercentPosition** tulajdonság használata előtt az objektumot az utolsó rekordra lépve fel kell tölteni. Az utolsó rekordra a **MoveLast** vagy a **FindLast** metódus használatával ugorhatunk. Ha a tulajdonságot a teljes feltöltés előtt használjuk, a pozíció a **RecordCount** tulajdonság által adott rekordszámhoz viszonyított lesz.

A **Recordset** objektumban nem javasolt az aktuális rekordról egy adott rekordra ugráshoz a **PercentPosition** tulajdonság használata, erre a célra megfelelőbb a **Bookmark** vagy az **AbsolutePosition** tulajdonság.

A **PercentPosition** tulajdonság egy értékre történt beállítását követően az értéknek megfelelő pozíció szerinti rekord válik aktuális rekorddá, és a tulajdonság ennek a rekordnak a közelítő pozícióját meghatározó értéket kapja.

A tulajdonság a csak előre görgethető pillanatfelvétel típusú kivételével mindhárom **Recordset** objektumtípuson használható, azonban a tábla típusú **Recordset** esetén a **PercentPosition** tulajdonság értéke mindig 50 lesz, ha a **Recordset** objektum **Index** tulajdonsága nem az érvényes értékre van beállítva. Megjegyzendő, hogy a tulajdonság nem érvényes az elavult **Dynaset**, **Snapshot** és **Table** objektumokon.

A **PercentPosition** tulajdonságot űrlapon vagy szövegbeviteli mezőn egy gördítősávval együtt használva a **Recordset** aktuális rekordjának pozíciójelzésére lehet használni.

A tulajdonság nem elérhető a csak előre gördíthető pillanatfelvétel típusú **Recordset**, vagy egy távoli adatbázisnak átadásra kerülő lekérdezéssel megnyitott **Recordset** objektum esetén.

Permissions

Szintaxis: *objektum*.**Permissions** [= *érték*]

A tulajdonság **Container** és **Document** objektumon érvényes, és az objektum **UserName** tulajdonságával adott felhasználói vagy csoportos jogosultságokat meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a jogosultságokat meghatározó szimbolikus konstans.

Bármely **Container** és **Document** objektumon az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	A jogosultság értelmezése:
dbSecNoAccess	Az objektumhoz nincs hozzáférés.
dbSecFullAccess	Teljes körű hozzáférés az objektumhoz.
dbSecDelete	Az objektum törölhető.
dbSecReadSec	Az objektum biztonsági információi olvashatók.
dbSecWriteSec	A hozzáférési jogosultságok módosíthatók.
dbSecWriteOver	Az **Owner** tulajdonság értéke módosítható.

A Tables **Container** és egy **Documents** halmazban lévő bármely **Document** esetén az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	A jogosultság értelmezése:
dbSecCreate	Új dokumentum hozható létre. (Csak egy **Container** objektummal érvényes.)
dbSecReadDef	A tábla meghatározás — az oszlop és index információkat is beleértve — olvasható.
dbSecWriteDef	A tábla meghatározás — az oszlop és index információkat is beleértve — módosítható vagy törölhető.
dbSecRetrieveData	A **Document** objektumból adatok olvashatók vissza.
dbSecInsertData	Rekordok hozzáadása engedélyezett.
dbSecReplaceData	Rekordok módosítása engedélyezett.
dbSecDeleteData	Rekordok törlése engedélyezett.

A Databases **Container** és egy **Documents** halmaz bármely **Document** objektumán az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	A jogosultság értelmezése:
dbSecDBCreate	Új adatbázis hozható létre. (Csak a Databases **Container** objektumon érvényes a rendszer adatbázisban [SYSTEM.MDW].)
dbSecDBExclusive	Kizárólagos hozzáférés.
dbSecDBOpen	Az adatbázis megnyitható.

Ezek a konstansok a DataAccess (DAO) objektumtárban az Object Browser által adott listában találhatók.

A tulajdonság a **Container** vagy **Document** objektum **UserName** tulajdonságában adott felhasználó írási/olvasási/módosítási jogosultságának meghatározásához használható.

A **Document** objektum örökli a **Container** objektumától annak **Inherit** tulajdonságában adott felhasználói jogosultságot. A **Document** objektum hozzáférési jellemzőit később a **Permission** és **UserName** tulajdonságainak beállításával lehet finomítani.

Picture

Szintaxis: *objektum*.**Picture** [= *érték*]

Ez a tulajdonság **Form** és **MDIForm** objektumon, valamint **Image**, **OLE** befogadó és **PictureBox** vezérlőelemen érvényes, és egy vezérlőelemben megjelenített grafika kiolvasására vagy beállítására használható. A tulajdonság **OLE** befogadó vezérlőelemen programfejlesztési fázisban nem elérhető és futásidőben csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy grafikát tartalmazó fájlt meghatározó karakteres kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha az *érték*:	Értelmezése:
(Nincs)	Nincs kép. (Alapértelmezett.)
(Bitkép, ikon, metafájl)	Egy grafikát ad meg. A grafika a programfejlesztési fázisban a Properties ablakból tölthető be. Futásidőben a tulajdonság a **LoadPicture** ( ) függvény bitképen, ikonon vagy metafájlon történő használatával állítható be.

A programfejlesztési fázisban a grafika a **Clipboard** objektummal vihető át, az Edit menü Copy, Cut és Paste parancsainak használatával. Futásidőben ugyanezen célra a **Clipboard** metódusai (**GetData** és **SetData**), és a nem szöveges **Clipboard** konstansok (**vbCFBitmap**, **vbCFMetafile** és **vbCFDIB**) esetében a **GetFormat** is érvényes.

A **Picture** tulajdonság programfejlesztési fázisban történő beállítása esetén a grafika az űrlappal együtt mentődik és töltődik be. Ha futtatható fájt hozunk létre, a fájl a képet is tartalmazza. A futásidőben betöltött képek nem kerülnek az alkalmazással együtt mentésre. Egy grafikát egy űrlapról vagy képmezőből fájlba mentésére a **SavePicture** utasítást kell használni.

Futásidőben a **Picture** tulajdonság bármely más objektum **DragIcon**, **Icon**, **Picture** és **Image** tulajdonságához, vagy a **LoadPicture**( ) függvény által visszaadott grafikához állítható be.

PID

Szintaxis: *objektum*.**PID** [= *érték*]

Ez a tulajdonság **Group** és **User** objektumon érvényes, és egy felhasználói vagy csoportos számla egyéni azonosítójának (Personal Identifier, PID) beállítására használható. A tulajdonság objektumgyűjteményhez még nem tartozó új objektum esetén csak olvasható, meglévő objektumok esetén nem hozzáférhető. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum 4-20 karakter hosszúságú **String** típusú kifejezés.

A **PID** tulajdonságot egy új **Group** objektum létrehozásakor a **Name** tulajdonsággal együtt, egy új **User** objektum létrehozásakor pedig a **Name** és **Password** tulajdonságokkal együtt kell beállítani.

Port

Szintaxis: *objektum*.**Port**

Ez a **Printer** objektumon érvényes tulajdonság a dokumentumnak a nyomtatóra küldéséhez használt kapu nevét adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A kap nevét, például LPT1:, az operációs rendszer határozza meg.

A **Printer** objektum tulajdonságainak hatása a nyomtató gyártója által adott vezérlőprogramtól függ: egyes tulajdonságok hatástalanok lehetnek vagy több tulajdonság azonos hatást eredményezhet. Erre vonatkozóan a nyomtató leírása tartalmazhat további információt.

PrevInstance

Szintaxis: *objektum*.**PrevInstance**

Ez az **App** objektumon érvényes tulajdonság az alkalmazás egy előző futó példányát meghatározó értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A tulajdonságot egy **Load** eseményeljárásban használhatjuk annak meghatározására, hogy a felhasználó már futtatja-e az alkalmazás egy példányát. Egyes alkalmazások esetén szükséges lehet, hogy a Windows operációs környezetben csak egy példánya futhasson.

Primary

Szintaxis: *objektum*.**Primary** [= *érték*]

Ez a tulajdonság **Index** objektumon érvényes, és az objektumnak egy tábla elsődleges indexekénti használatát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság objektumgyűjteményhez még nem tartozó új **Index** objektum esetén írható és olvasható, míg a meglévő, **Indexes** halmazban lévő objektum esetén csak olvasható. Ha az **Index** objektum **TableDef** objektumhoz csatolt, de az még nem tartozik **TableDefs** objektumgyűjteményhez, a tulajdonság írható és olvasható is lesz. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy elsődleges indexet meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True	Az **Index** objektum elsődleges indexet jelöl.
False	Az **Index** objektum nem elsődleges indexet jelöl.

Az elsődleges index egy vagy több mezőt tartalmaz, amely előre adott sorrendben és egyedileg azonosít minden rekordot egy táblában. Mivel az indexmező egyedi, az **Index** objektum **Unique** tulajdonsága **True** értékű. Ha az elsődleges index egynél több mezőt tartalmaz, az egyes mezők tartalmazhatnak kettős értékeket is, azonban az értékek minden egyes kombinációjának egyedinek kell lennie. Az elsődleges index egy kulcsot tartalmaz a táblához és rendszerint az elsődleges kulccsal azonos mezőket tartalmazza.

A táblákhoz nem kötelező indexeket létrehozni, azonban másként a nagyméretű, nem indexelt tábla adott rekordjához történő hozzáférés sok időt igényel. Az **Index** objektumban minden egyes **Field** objektum **Attributes** tulajdonsága meghatározza a rekordok sorrendjét, és ebből következően az index elérési módját. Az adatbázisunkban egy új tábla létrehozásakor célszerű egy vagy több mezőn azok minden egyes rekordját egyedileg azonosító indexelést is létrehozni, majd az **Index** objektum **Primary** tulajdonságát **True** értékre állítani.

Egy táblához elsődleges indexet beállítva, az automatikusan a tábla elsődleges indexévé is válik.

PrinterDefault

Szintaxis: *objektum*.**PrinterDefault** [= *érték*]

Ez a tulajdonság **CommonDialog** vezérlőelemen érvényes, és a Print párbeszédpanelen végzett felhasználói kiválasztásokkal az alapértelmezett nyomtató beállításainak módosítását meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a rendszer alapértelmezett nyomtatójának a beállításokkal történt módosítását meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A felhasználó által a Print párbeszédpanel Setup részében végrehajtott kiválasztások (nyomtató kiválasztása, orientáció, papírtovábbítás stb.) a nyomtató beállításainak a felhasználói WIN.INI fájlban (a Windows NT esetén a Registry fájlban) történő módosítására használhatók.

False A felhasználó által végrehajtott kiválasztások nem használhatók az alapértelmezett nyomtató beállításainak módosítására.

Ha a **PrinterDefault** tulajdonság értéke **True**, a Visual Basic **Printer** objektumába közvetlenül nyomtató programkód írható, egyébként GDI (Graphic Device Interface) hívásokat kell használni a vezérlőelem **hDC** tulajdonságában adott nyomtató használatához.

Ha előzőleg már történt a **Printer** objektumba nyomtatás, a beállítás előtt a `Prin-`

`ter.EndDoc` használatával ellenőrizni kell, hogy a nyomtatási tevékenység befejeződött-e. Ezzel felszabadul a nyomtatóhoz kapcsolódó **hDC**, és így a következő **Printer** objektumba történő nyomtatásnál új **hDC** tulajdonság kapható. Az ellenőrzés elmulasztása azzal a veszéllyel járhat, hogy a felhasználó egy új nyomtatót választ, mialatt a **Printer** objektum még a régi nyomtatóra vonatkozó kezelőt tartalmazza.

PrintQuality

Szintaxis: *objektum*.**PrintQuality** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Printer** objektumon érvényes, és a nyomtató felbontását meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a nyomtató felbontását meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	Értéke:	A nyomtató felbontása:
dbPRPQDraft	-1	Piszkozat minőség.
dbPRPQLow	-2	Alacsony felbontás.
dbPRPQMedium	-3	Közepes felbontás.
dbPRPQHigh	-4	Nagy felbontás.

Ezek a konstansok a Visual Basic objektumtárban az Object Browser által adott listában találhatók. Az előre definiált negatív értékek helyett a felbontást dpi (pont/inch) egységben, pozitív értékként is megadhatjuk.

Az alapértelmezett érték és a beállítás hatása a nyomtatótól és a vezérlőprogramtól függ, így egyes nyomtatóknál az adott beállítások közül néhány vagy akár az összes is ugyanazon hatású lehet.

A **Printer** objektum tulajdonságainak hatása a nyomtató gyártója által adott vezérlőprogramtól függ: egyes tulajdonságok hatástalanok lehetnek vagy több tulajdonság azonos hatást eredményezhet. Erre vonatkozóan a nyomtató leírása tartalmazhat további információt.

ProductName

Szintaxis: *objektum*.**ProductName**

Ez az **App** objektumon érvényes tulajdonság a futó alkalmazás gyártmánynevét tartalmazó karakterláncot ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója

egy objektumot kijelölő objektumkifejezés.

A tulajdonság értékét az EXE Options párbeszédpanelen lehet beállítani a programfejlesztési fázisban.

Public

Szintaxis:

Ez a futásidőben nem elérhető tulajdonság **ClassModule** objektumon érvényes, és egy általunk létrehozott osztálynak a projekten kívüli láthatóságát meghatározó érték beállítására használható.

A **Public** tulajdonság lehetséges értékei és értelmezésük a következő:

True Az általunk létrehozott osztály a projekten belül és a projekten kívüli alkalmazások számára is látható.

False Az általunk létrehozott osztály csak a projekten belül látható. (Alapértelmezett.)

A **Public** és **Instancing** tulajdonságok együttes használatával az általunk definiált osztályok más alkalmazások számára való láthatósága a következők szerint határozható meg:

Public	Instancing	A láthatóság:
False	Tetszőleges	Az osztály csak az azt definiáló projekten belül hozható létre. Az osztály nem látható más alkalmazások számára.
True	0	Az osztály csak az azt definiáló projekten belül hozható létre. Az osztály más alkalmazások számára is látható és így vezérelhető is, amint az osztály egy példánya létezik.
True	1	Az osztály példányai a projekten belül és kívül is létrehozhatók. A projekten kívüli OLE ügyfélnek az osztály példányára vonatkozó minden egyes kérése az OLE kiszolgáló egy különálló, újabb példányának indítását eredményezi. Az osztály más alkalmazások számára is látható és így vezérelhető is, amint az osztály egy példánya létezik.
True	2	Az osztály példányai a projekten belül és kívül is létrehozhatók. A projekten kívüli OLE ügyfélnek az osztály példányára vonatkozó minden egyes kérése az OLE kiszolgáló egy már futó példányának adódik át. Ha az OLE kiszolgálónak nincs futó példánya, az osztályt szolgáltató másolat készül. Az osztály más alkalmazások számára is látható és így vezérelhető is, amint az osztály egy példánya létezik.

QueryTimeout

Szintaxis: *objektum*.**QueryTimeout** [= *érték*]

A tulajdonság **Database** objektumon érvényes, és egy lekérdezés ODBC adatbázison történő végrehajtása során a Microsoft Jet hibajelzése előtti várakozási idejének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a Microsoft Jet adatbázis-szerkesztőnek a várakozási idő túllépés hiba kiadása előtt szükséges idő másodpercekben adott, **Integer** típusú értéke.

Egy ODBC — például SQL Server — adatbázis használata során a hálózati forgalom vagy a kiszolgáló leterheltsége miatt késlekedés léphet fel. A határozatlan ideig tartó várakozás helyett megadhatjuk a Microsoft Jet hibajelzés előtti várakozási idejét.

A **QueryTimeout** tulajdonságot egy **Database** objektumon használva egy globális várakozási érték állítható be az adatbázissal kapcsolatos minden lekérdezésre.

A Microsoft Jet hibajelzés kiadása előtti alapértelmezetten bármelyik .INI fájlbeli beállítástól függetlenül 60 másodperces várakozási ideje helyett más időtartam is megadható. A **Database** objektumon a **QueryTimeout** tulajdonság beállítása felülírja az alapértelmezett értéket.

ReadOnly

Szintaxis: *objektum*.**ReadOnly** [= *érték*]

A tulajdonság a **Data** vezérlőelemen a vezérlőelem **Database** objektumának csak olvasható hozzáférésre történt megnyitását előíró érték, a **FileListBox** vezérlőelemen pedig a listában csak olvasható attribútumú fájlok megjelenítését előíró érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az írási/olvasási hozzáférést meghatározó **Boolean** típusú kifejezés.

A **ReadOnly** tulajdonság lehetséges értékei és értelmezésük a következő:

True **Data** vezérlőelemen — A vezérlőelem **Database** objektum csak olvasható hozzáférésre van megnyitva, az adatok módosítása nem engedélyezett.
FileListBox vezérlőelemen — A vezérlőelem listája csak olvasható fájlokat tartalmaz.

False **Data** vezérlőelemen — A vezérlőelem **Database** objektum írható és olvasható hozzáférésre van megnyitva. (Alapértelmezett.)
FileListBox vezérlőelemen — A vezérlőelem listája nem tartalmazza a csak olvasható fájlokat. (Alapértelmezett.)

A **ReadOnly** tulajdonság használatával a **Data** vezérlőelemen előírható, hogy az alapjául szolgáló **Database** objektumban az adatok módosíthatók-e. A csak olvasható **Database** egyben gyorsabb adathozzáférést is biztosít. A tulajdonságot csak az adatbázisnak az alkalmazásunk általi első megnyitásakor kell használni. Ha az alkalmazásunk előzőleg az adatbázis másik példányait már megnyitotta, a tulajdonság hatástalan lesz. A **ReadOnly** tulajdonság módosítását az adatbázis minden példányának bezárásával és a **Refresh** metódus használatával lehet érvényesíteni.

A Professional Edition esetén ez a tulajdonság az **OpenDatabase** metódus *csak_olvasható* argumentumának felel meg.

A **ReadOnly** tulajdonságot a **FileListBox** vezérlőelemen használva az operációs környezetben használt standard fájlattribútumoknak alapján a megjelenítendő fájlok típusa írható elő. A tulajdonság futásidejű beállítása újra beállítja a megfelelő attribútumú fájlok megjelenítését.

RecordCount

Szintaxis: *objektum*.**RecordCount**

Ez a csak olvasható, a **Recordset** és **TableDef** (valamint az elavult **Dynaset**, **Snapshot** és **Table**) objektumon érvényes tulajdonság az objektumban használt rekordok számát **Long** típusú értékként adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **RecordCount** tulajdonsággal egy **Recordset** vagy **TableDef** objektum eddig már használt rekordjainak — tehát nem az objektum összes rekordjainak — számát lehet meghatározni. Ha már az utolsó rekordot is használtuk, a **RecordCount** értéke megegyezik az objektum nem törölt összes rekordjainak számával. Az utolsó rekord használatát a **Recordset** objektumon a **MoveLast** vagy **FindLast** metódusokkal lehet kikényszeríteni. A lekérdezés által visszaadott rekordok közelítő száma az SQL **Count( )** függvény használatával is meghatározható.

Ha az alkalmazásunk rekordot töröl egy eredményhalmaz típusú **Recordset** (vagy az elavult **Dynaset**) objektumból, a **RecordCount** értéke csökken, azonban a más felhasználók által végzett rekordtörlés csak akkor módosítja a tulajdonság értékét, ha az aktuális rekord a törölt rekordra pozícionált lesz. Ha olyan tranzakciót hajtunk végre, amely módosítja a **RecordCount** értékét, majd a tranzakciót visszavonjuk, a tulajdonság értéke nem tükrözi a megmaradó rekordok aktuális számát.

Egy pillanatfelvétel típusú **Recordset** (vagy az elavult **Snapshot**) objektumban az alapjául szolgáló táblák változásai nem hatnak a **RecordCount** értékére.

Egy tábla típusú **Recordset**, **TableDef** (vagy az elavult **Table**) objektumban a **RecordCount** értéke mindig a tábla rekordjainak értékét adja meg, és bármelyik felhasználó általi rekordtörlés vagy hozzáadás a tulajdonság értékét azonnal módosítja.

Rekordot nem tartalmazó **Recordset** vagy **TableDef** objektum **RecordCount** tulajdonságának értéke 0. A csatolt táblákkal vagy ODBC adatbázisokkal végzett munkánál a **RecordCount** tulajdonság értéke mindig -1 lesz.

A **Recordset** objektumon használt **Requery** metódus a **RecordCount** tulajdonság értékét ugyanúgy ismét beállítja, mintha a lekérdezés ismételten lett volna végrehajtva.

RecordsAffected

Szintaxis: *objektum*.**RecordsAffected**

Ez a **Database** és **QueryDef** objektumon érvényes tulajdonság az utolsóként végrehajtott **Execute** metódus által érintett rekordok számát **Long** típusú értékként adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **RecordsAffected** tulajdonság értéke a módosító lekérdezés futása során törölt, frissített vagy beillesztett rekordok számát tartalmazza.

RecordSelectors

Szintaxis: *objektum*.**RecordSelectors** [= *érték*]

A tulajdonság **DBGrid** vezérlőelemen érvényes, és a vezérlőelemen a rekordkiválasztók megjelenítését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a rekordválasztók megjelenítését meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True	A rekordkiválasztók megjelennek a **DBGrid** vezérlőelemen.
False	A rekordkiválasztók nem jelennek meg a **DBGrid** vezérlőelemen.

A rekordkiválasztók a **DBGrid** vezérlőelem sorainak bal oldalán jelennek meg, és amikor a felhasználó a rekordválasztót kijelölté teszi, ezzel a teljes rekord, azaz a **DBGrid** adott sora is kiválasztottá válik.

Recordset

Szintaxis: *objektum*.**Recordset** [= *érték*]

A tulajdonság **Data** vezérlőelemen érvényes, és a vezérlőelem tulajdonságaival vagy egy létező **Recordset** objektummal megadott **Recordset** objektum kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **Recordset** objektumot megadó karakteres kifejezés.

A **Data** vezérlőelem az alkalmazásunk elindulásakor automatikusan inicializálódik. Ha a **Connect**, **DatabaseName**, **Options**, **RecordSource**, **Exclusive**, **ReadOnly** és **RecordsetType** tulajdonságok érvényesek, vagy ha ezek **Data** vezérlőelem tulajdonságai futásidőben beállítottak és a **Refresh** metódust használtuk, a Microsoft Jet adatbázis-szerkesztő megkísérli ezen tulajdonságok alapján egy új **Recordset** objektum létrehozását. Az új **Recordset** a **Data** vezérlőelem **Recordset** tulajdonságán keresztül lesz hozzáférhető. A **Recordset** tulajdonságot bármely más **Recordset** objektumhoz hasonlóan lehet használni.

Ha szükséges, a **Recordset** típusát a **Data** vezérlőelem **RecordsetType** tulajdonságával lehet az alapértelmezetten eredményhalmaz típus helyett pillanatfelvétel vagy tábla típusra beállítani. Ha a Jet nem képes a kért típust létrehozni, kezelhető hiba következik be.

A létrehozott **Recordset** típusát futásidőben a **Recordset** tulajdonság esetén a **Type** tulajdonság, **Data** vezérlőelem esetén pedig a **RecordsetType** tulajdonság ellenőrzésével lehet meghatározni.

A **Recordset** akkor sem frissíthető, ha eredményhalmaz vagy tábla típusú **Recordset** kérésünk volt. Ha az alapjául szolgáló tábla vagy mező nem frissíthető, a **Recordset** részben vagy egészében csak olvasható lesz. A rekordok programkódból történő módosíthatóságát a **Database** és **Recordset** objektumok esetén az **Updatable** tulajdonság, a **Field** objektum esetén a **DataUpdatable** tulajdonság vizsgálatával lehet meghatározni.

A **Recordset** által visszaadott rekordok számának meghatározásához a **MoveLast** vagy **FindLast** metódussal az objektum utolsó rekordjára kell ugorni, majd meg kell vizsgálni a **Recordset** objektum **RecordCount** tulajdonságának értékét.

Ha akár programkódból, akár másik **Data** vezérlőelem használatával hoztunk létre **Recordset** objektumot, a **Data** vezérlőelem **Recordset** tulajdonságát erre az új objektumra lehet beállítani. A **Data** vezérlőelemben bármely létező **Recordset** és az azzal kapcsolatos **Database** objektum az új **Recordset** objektumnak a **Recordset** tulajdonsághoz történt beállításakor felszabadul.

Ha a **Recordset** tulajdonság beállított, a **Data** vezérlőelem az aktuális **Recordset**

vagy **Database** objektumot csak felszabadítja, de nem zárja be. Ha nincs más felhasználó, az adatbázis automatikusan bezáródik. A **Recordset** tulajdonság beállítását megelőzően megfontolható a **Data** vezérlőelemmel kapcsolatos **Recordset** vagy **Database** bezárása.

A **Data** vezérlőelemhez kötött vezérlőelemek **DataField** tulajdonságának az új **Recordset** objektum mezőneveihez kell illenie.

A **Data** vezérlőelemmel létrehozott mindegyik **Recordset** a **Workspace**(0) objektumba kerül. Ha a vezérlőelemet másik **Workspace** objektumbeli adatbázis kezelésére akarjuk használni, az adatbázist meg kell nyitni, egy új **Recordset** objektumot kell létrehozni, majd a **Data** vezérlőelem **Recordset** tulajdonságát erre az új objektumra kell beállítani.

A **Data** vezérlőelem **Recordset** objektumának tulajdonságaira a **Recordset** tulajdonságát használva mindig tudunk hivatkozni. A **Recordset** objektumra történő közvetlen hivatkozással a **Table** objektumokkal együtt az **Index**, egy **QueryDef** **Parameters** halmaza vagy a **Recordset** típusa határozható meg.

RecordsetType

Szintaxis: *objektum*.**RecordsetType** [= *érték*]

Ez a tulajdonság **Data** vezérlőelemen érvényes, és vezérlőelem által létrehozandó **Recordset** objektum típusát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a létrehozandó **Recordset** objektum típusát meghatározó szimbolikus konstans vagy **Integer** típusú érték. (Ez az argumentum nem azonos a **Recordset** objektumtípusának azonosítójával!)

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha a konstans:	Értéke:	A létrehozandó **Recordset** típusa:
vbRSTTypeTable	0	Tábla típusú **Recordset**.
vbRSTTypeDynaset	1	Eredményhalmaz típusú **Recordset**.
vbRSTTypeSnapshot	2	Pillanatfelvétel típusú **Recordset**.

Ha szükséges, a **RecordsetType** tulajdonsággal lehetséges a Microsoft Jet által létrehozandó **Recordset** típusát eredményhalmaz, pillanatfelvétel vagy tábla típusra beállítani. Ha a Jet nem képes a kért típust létrehozni, kezelhető hiba következik be.

Ha a **Recordset** objektum **Data** vezérlőelem általi létrehozása előtt nincs meghatározva a **RecordsetType** tulajdonság, az alapértelmezett eredményhalmaz típusú

Recordset jön létre.

Ha a **Recordset** objektum a **Data** vezérlőelem nélkül (akár egy másik Data használatával) volt létrehozva, majd a **Recordset** tulajdonság ezzel az új objektummal van beállítva, a **Data** vezérlőelem **Recordset** tulajdonsága az új objektum **Recordset.Type** tulajdonságát kapja.

Bár a **Data** vezérlőelem nem támogatja az elavult **Dynaset** objektumot, azonban alapértelmezetten azzal kompatíbilis eredményhalmaz típusú **Recordset** objektumot hoz létre, így a meglévő programkódot nem kell módosítani.

RecordSource

Szintaxis: *objektum*.**RecordSource** [= *érték*]

Ez a tulajdonság **Data** vezérlőelemen érvényes, és vezérlőelemhez az alapjául szolgáló tábla, SQL utasítás vagy **QueryDef** objektum kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a nevet meghatározó karakteres kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha az *érték*:	Értelmezése:
Egy táblanév	A **Database** objektum **TableDefs** objektumgyűjteményében lévő táblák egyikének neve.
Egy SQL lekérdezés	Egy érvényes, az adatbázisnak megfelelő szintaxisú SQL karakterlánc.
Egy **QueryDef**	A **Database** objektum **QueryDefs** objektumgyűjteményében lévő **QueryDef** objektumok egyikének neve.

A **RecordSource** tulajdonság az űrlap kötött vezérlőelemén keresztül elérhető adatok forrásának meghatározására használható.

Ha a tulajdonságot az adatbázis egy létező táblájának nevéhez állítjuk be, a **Data** vezérlőelemhez csatolt kötött vezérlőelemek számára a tábla mindegyik mezője látható lesz. A kiolvasott rekordok sorrendjét a **Recordset Index** tulajdonságának használatával kiválasztott **Index** objektum állítja be. Ha az **Index** tulajdonság nem beállított, az adatokat nem előírt sorrendben kapjuk vissza.

Ha a **RecordSource** tulajdonságot az adatbázis egy létező **QueryDef** objektumának nevére állítjuk be, a **Data** vezérlőelemhez csatolt kötött vezérlőelemek számára a **QueryDef** által visszaadott mindegyik mező látható lesz. A kiolvasott rekordok sorrendjét a **QueryDef** objektum állítja be. Ha a **QueryDef** nem határoz meg sorrendet, az

Ha a **RecordSource** tulajdonságot rekordokat visszaadó SQL utasítással állítjuk be, a **Data** vezérlőelemhez csatolt kötött vezérlőelemek számára az SQL utasítás által visszaadott mindegyik mező látható lesz. Az SQL utasításban a **Data** vezérlőelemmel létrehozott **Recordset** által visszaadott rekordok sorrendjének módosítására **ORDER BY** záradékot, illetve a rekordok szűréséhez **WHERE** záradékot lehet használni. Ha a **Database** és **Connect** tulajdonságban adott adatbázis nem Jet adatbázis, vagy ha az **Options** tulajdonságban **dbSQLPassThrough** argumentum van beállítva, a használt adatbázisnak megfelelő szintaxisú SQL lekérdezést kell használni.

Egy kifejezésből a **QueryDef** vagy SQL utasítás által történő értékvisszaadáskor a kifejezés mezőnevét a Microsoft Jet automatikusan hozza létre. Általában a név `Expr1`*nnn* alakú, ahol *nnn* egy 000 kezdetű sorszám, azonban a legtöbb esetben a kötött vezérlőelem oszlopnevét jelző nevű kifejezést akarunk használni. Ilyen esetben az SQL **SELECT** utasítás **AS** záradékú alakját kell használni.

A **RecordSource** tulajdonság futásidejű módosítását egy **Refresh** metódusnak kell követnie az új érték érvényesítéséhez.

Mindegyik kötött vezérlőelem **DataField** tulajdonságának érvényes beállításúnak kell lennie. A **Data** vezérlőelem **RecordSource** tulajdonságának módosítása, majd a **Refresh** metódus használata után a **Recordset** az új objektumot határozza meg, ezzel a kötött vezérlőelemek **DataField** tulajdonságát érvénytelenné téve és belsőleg figyelt hibát okozva.

Replicable

Szintaxis: *objektum*.**Replicable** [= *érték*]

A tulajdonság **Database** objektumon érvényes, és adatbázison az adatbázishoz mezők, táblák és tulajdonságok hozzáadását és az adatbázis többszörözhetőségét, az objektumokon pedig egy objektum alappéldányban történt létrehozása vagy abba importálása utáni többszörözhetőségét meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az adatbázis vagy objektum többszörözhetőségét meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True	Az adatbázis vagy lokális objektum többszörözhető.
False	Az adatbázis vagy lokális objektum nem többszörözhető. (Alapértelmezett.)

Egy adatbázis esetén a **True** értékű tulajdonság már nem módosítható, a módosítási kísérlet hibaüzenetet eredményez.

Egy új objektum (tábla, lekérdezés, űrlap, jelentés, makró vagy modul) létrehozásakor a lehet példány, amelyet lokális helyett többszörözhetővé kívánunk tenni. Ehhez

vagy az alappéldányban kell létrehozni, vagy abba kell importálni azt, majd ha már az alappéldány részévé vált, az objektum **Replicable** tulajdonságát **True** értékre kell beállítani.

Az az objektum, amelynek **KeepLocal** vagy **Replicable** tulajdonságát be kívánjuk állítani, ezeket a tulajdonságokat más objektumtól örökölhette, azonban a tulajdonságok értékeit nem, azt minden egyes objektumra közvetlenül kell beállítani.

ReplicaID

Szintaxis: *objektum*.**ReplicaID**

A tulajdonság **Database** objektumon érvényes, és a másolatot szolgáltató, a Microsoft Jet által előállított 16-bites egyedi azonosító (OLE 2 GUID) értékét adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **ReplicaID** a **MasterDesignID** tulajdonsággal együtt a másolathalmaz alappéldányának módosítására használható.

Required

Szintaxis: *objektum*.**Required** [= *érték*]

A tulajdonság **Field** és **Index** objektumon érvényes, és az objektum által csak nem **Null** érték elfogadását meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság objektumgyűjteményhez még nem tartozó új objektum esetén írható és olvasható, míg a **Recordset** vagy **TableDef** objektumbeli **Indexes** objektumgyűjteményhez tartozó **Index** objektumoknál csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **Null** érték objektum általi elfogadását meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A **Null** érték nem megengedett.
False A **Null** érték megengedett.

Field objektumnál a **Required** tulajdonság használata attól a **Fields** halmazt tartalmazó objektumtól függ, amelyhez a **Field** tartozik:

Ha az objektum:	A tulajdonság használata:
Index	Nem támogatott.
QueryDef	Csak olvasható.
Recordset	Csak olvasható.

Relation	Nem támogatott.
TableDef	Írható és olvasható.

Egy **Field** objektum esetén a **Required** tulajdonságot az **AllowZeroLength**, **ValidateOnSet** vagy **ValidationRule** tulajdonságokkal együtt használva a **Field** objektum **Value** tulajdonságának érvényessége ellenőrizhető. Ha a **Required** értéke **False**, a mező **Null**, továbbá az **AllowZeroLength** és **ValidationRule** tulajdonságok beállításának megfelelő értékeket tartalmazhat.

Amikor a **Required** tulajdonság **Index** és **Field** objektumhoz is beállítható, a **Field** objektumhoz kell beállítani. A tulajdonság érvényessége a **Field** objektumhoz történő beállítás során ellenőrzésre kerül, mielőtt azt egy **Index** objektum kapná.

Restartable

Szintaxis: *objektum*.**Restartable**

A tulajdonság **Recordset** objektumon érvényes, és a **Requery** metódus, azaz az objektum alapjául szolgáló lekérdezés újbóli végrehajtásának a **Recordset** általi támogatását jelző értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Restartable** lehetséges értékei és értelmezésük a következő:

True	A **Recordset** támogatja a **Requery** metódust.
False	A **Recordset** nem támogatja a **Requery** metódust. A tábla típusú **Recordset** esetén csak ez érvényes.

A **Restartable** tulajdonságot a **Requery** metódusnak egy **Recordset** objektumon történő használata előtt ellenőrizni kell. Ha annak értéke **False**, a **Recordset** alapjául szolgáló **QueryDef** objektumon a lekérdezés ismételt végrehajtásához az **OpenRecordset** metódust kell használni.

A **Requery** metódust a paraméterek értékének ellenőrzése után egy **Recordset** objektum alapjául szolgáló paraméterlekérdezés frissítéséhez lehet használni.

ReturnsRecords

Szintaxis: *objektum*.**ReturnsRecords** [= *érték*]

Ez a tulajdonság **QueryDef** objektumon érvényes, és egy külső adatbázis SQL utasítást átadó lekérdezésének rekordvisszaadását meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a rekord visszaadá-

sát meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True Az utasítást átadó lekérdezés rekordokat ad vissza. (Alapértelmezett.)
False Az utasítást átadó lekérdezés nem ad vissza rekordokat.

A külső adatbázisok SQL utasítást átadó lekérdezései közül egyesek — például egy SQL UPDATE — nem adnak vissza rekordokat. Ha a lekérdezés rekordokat ad vissza, a **ReturnsRecords** tulajdonságot **True** értékre kell beállítani.

A **Connect** tulajdonság beállításának meg kell előznie a **ReturnsRecords** beállítását.

Revision

Szintaxis: *objektum*.**Revision**

Ez a futásidőben csak olvasható tulajdonság **App** objektumon érvényes, és a projekt javítási verziószámát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Revision** tulajdonság értéke 0 és 999 közötti szám lehet.

A tulajdonság a futó alkalmazás verziójára vonatkozó információt ad vissza. A tulajdonság beállítása a programfejlesztési fázisban az EXE Options párbeszédpanelen lehetséges.

Row

Szintaxis: *objektum*.**Row** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **DBGrid** és **Grid** vezérlőelemen érvényes, és a vezérlőelem aktív cellájának meghatározására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az aktív cellát tartalmazó sor száma.

A tulajdonságot a **Grid** vezérlőelemben az aktív cella kijelölésére vagy annak meghatározására használhatjuk, hogy az aktív cella melyik sorban található. A sorok számozása nullától, azaz a legfelső sortól kezdődik. A tulajdonság futásidejű beállítása nem módosítja, hogy mely cellák kiválasztottak. Egy cellatartomány beállítására a **SelStartCol**, **SelEndCol**, **SelStartRow** és **SelEndRow** tulajdonságokat lehet használni.

RowDividerStyle

Szintaxis: *objektum*.**RowDividerStyle** [= *érték*]

Ez a tulajdonság **DBGrid** vezérlőelemen érvényes, és a vezérlőelem sorai közötti határvonal stílusát meghatározó érték kiolvasására vagy beállítására alkalmas. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a határvonal stílusát meghatározó **Integer** típusú szám.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha az *érték*:	A határvonal:
0	Nincs határvonal.
1	Fekete vonal.
2	Sötétszürke vonal. (Alapértelmezett.)
3	Kiemelt hatású vonal.
4	Besüllyesztett hatású vonal.
5	A határvonal a **ForeColor** tulajdonság szerinti színű.

A **RowDividerStyle** tulajdonság nem hat a határvonal elhúzhatóságára. Amikor a határvonal kiemelt vagy besüllyesztett hatású, a Microsoft Windows állítja be a színeket.

RowHeight

Szintaxis: *objektum*.**RowHeight**[(*szám*)] [= *érték*]

Ez a tulajdonság **DBGrid** és **Grid** vezérlőelemeken érvényes, és **DBGrid** esetén a vezérlőelemben lévő összes sor magasságát a befogadóra érvényes mértékegységben, a **Grid** esetén pedig az adott sor (rekord) magasságát twip egységben meghatározó érték kiolvasására vagy beállítására alkalmas. A tulajdonság **Grid** vezérlőelemen a programfejlesztési fázisban nem elérhető. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A *szám* argumentum a **Grid** vezérlőelem sorának számát megadó érték.
- A tulajdonság beállításához használható *érték* argumentum a magasságot meghatározó numerikus kifejezés.

A **Grid** vezérlőelemen a **RowHeight** legkisebb értéke 1 (képpont) lehet. A felhasználó bármely sor **RowHeight** tulajdonságát futásidőben a sorok közötti rácsvonal egérrel történő elhúzásával is módosíthatja.

RowIsVisible

Szintaxis: *objektum*.**RowIsVisible**(*index*)

Ez a tulajdonság **Grid** vezérlőelemen érvényes, és egy adott sornak a vezérlőelembeni láthatóságát meghatározó értéket ad vissza. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *index* argumentum a vezérlőelem egy sorát meghatározó **Integer** típusú kifejezés. Az *index* argumentum értéke 0 és a **Rows**-1 közötti szám lehet.

A **RowIsVisible** lehetséges értékei és értelmezésük a következő:

True	Az adott sor éppen látható.
False	Az adott sor éppen nem látható.

RowPos

Szintaxis: *objektum*.**RowPos**(*index*)

Ez a tulajdonság **Grid** vezérlőelemen érvényes, és egy adott sor bal felső sarka és a vezérlőelem bal felső sarka közötti távolságot meghatározó értéket ad vissza. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *index* argumentum a vezérlőelem egy sorát meghatározó **Integer** típusú kifejezés. Az *index* argumentum értéke 0 és a **Rows**-1 közötti szám lehet.

A **RowPos** által visszaadott érték twip egységben értendő. Egy (logikai) cm = 567 twip.

Rows

Szintaxis: *objektum*.**Rows** [= *érték*]

Ez a tulajdonság **Grid** vezérlőelemen érvényes, és használatával a vezérlőelem összes sorának számát lehet kiolvasni vagy beállítani. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az összes sor 1 és 2000 közötti megengedett száma.

A tulajdonsággal futásidőben dinamikusan megnövelhető a **Grid** vezérlőelem mi-

nimálisan egy oszlopból és egy sorból álló mérete.

RowSource

Szintaxis:

Ez a futásidőben nem elérhető tulajdonság **DBCombo** és **DBList** vezérlőelemen érvényes, és a vezérlőelem listamezőjének felöltésére használandó **Data** vezérlőelemet meghatározó értéket határozza meg. A **DBCombo** és **DBList** vezérlőelem listamezőjének felöltéséhez a **RowSource** tulajdonságot a programfejlesztés fázisában a Properties ablakban kell beállítani.

A mezőkapcsolatnak a **Recordset** objektumban a **Data** vezérlőelem általi teljessé tételéhez egy **Field** objektum nevét is meg kell adni a **ListField** vagy **BoundColumn** tulajdonságokban. Eltérően a **ListField** tulajdonságtól, a **RowSource** futásidőben nem érhető el.

ScaleHeight

Szintaxis: *objektum*.**ScaleHeight** [= *érték*]

Ez a tulajdonság **Form**, **MDIForm** és **Printer** objektumon, továbbá **PictureBox** vezérlőelemen érvényes, és használatával a grafikus metódusok használatakor vagy egy vezérlőelem pozicionálásakor az objektumbelső függőleges egységeinek száma olvasható ki vagy állítható be. Az **MDIForm** objektum esetén a tulajdonság a programfejlesztési fázisban nem elérhető, futásidőben pedig csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a függőleges méretet meghatározó numerikus kifejezés. Ha a kifejezés negatív értékű, a koordinátarendszer a fentről lefelé növekvő helyett lentről felfelé növekvő irányítottságú lesz.

A tulajdonság a rajzoláshoz vagy nyomtatáshoz saját, relatív koordinátaskála készítésére használható, például a `ScaleHeight = 100` beállítás 100 egységnyi magasságot, illetve a magasság 1/100 részének megfelelő függőleges egységet állít be. A standard mértékegységek (mm, inch, twip, pont stb) a **ScaleMode** tulajdonsággal definiálhatók. A **ScaleHeight** és **ScaleWidth** nem azonos a **Height** és **Width** tulajdonsággal!

A **ScaleHeight**, **ScaleWidth**, **ScaleLeft** és **ScaleTop** tulajdonságok együttesen a felhasználói igényeknek megfelelő, pozitív és negatív koordinátákkal rendelkező koordinátarendszert határoznak meg, és a **ScaleMode** tulajdonsággal a következőképpen hatnak egymásra:

- Bármely másik **Scale** tulajdonságot bármely értékre állítva a **ScaleMode** tulaj-

donság értéke 0 lesz.

- A **ScaleMode** értékét bármely nullánál nagyobb értékre állítva a **ScaleHeight** és **ScaleWidth** értéke az új mértékegység, a **ScaleLeft** és **ScaleTop** értéke pedig 0 lesz.

A **ScaleHeight**, **ScaleWidth**, **ScaleLeft** és **ScaleTop** tulajdonságok értékeit a **Scale** metódussal egyetlen sorban is be lehet állítani.

Az **MDIForm** objektum esetén a **ScaleHeight** csak arra a területre vonatkozik, amely az űrlapon nincs egy **PictureBox** vezérlőelemmel takarva, ezért kerülni kell a tulajdonság **PictureBox** vezérlőelemnek egy **Resize** eseményeljárásbeli méretezésére történő használatát.

ScaleLeft

Szintaxis: *objektum*.**ScaleLeft** [= *érték*]

Ez a tulajdonság **Form** és **Printer** objektumon, továbbá **PictureBox** vezérlőelemen érvényes, és használatával a grafikus metódusok használatakor vagy egy vezérlőelem pozicionálásakor az objektum bal szélső élének vízszintes koordinátája olvasható ki vagy állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a vízszintes koordináta értékét meghatározó numerikus kifejezés.

A **ScaleHeight**, **ScaleWidth**, **ScaleLeft** és **ScaleTop** tulajdonságok együttesen a felhasználói igényeknek megfelelő, pozitív és negatív koordinátákkal rendelkező koordinátarendszert határoznak meg, és a **ScaleMode** tulajdonsággal a következőképpen hatnak egymásra:

- Bármely másik **Scale** tulajdonságot bármely értékre állítva a **ScaleMode** tulajdonság értéke 0 lesz.
- A **ScaleMode** értékét bármely nullánál nagyobb értékre állítva a **ScaleHeight** és **ScaleWidth** értéke az új mértékegység, a **ScaleLeft** és **ScaleTop** értéke pedig 0 lesz. A **CurrentX** és **CurrentY** tulajdonságok értékei az aktuális pont új koordinátáinak megfelelően módosulnak.

A **ScaleHeight**, **ScaleWidth**, **ScaleLeft** és **ScaleTop** tulajdonságok értékeit a **Scale** metódussal egyetlen sorban is be lehet állítani.

A **ScaleLeft** és **ScaleTop** nem azonos a **Left** és **Top** tulajdonsággal!

ScaleMode

Szintaxis: *objektum*.**ScaleLeft** [= *érték*]

Ez a tulajdonság **Form** és **Printer** objektumon, továbbá **PictureBox** vezérlőelemen érvényes, és használatával a grafikus metódusok használatakor vagy egy vezérlőelem pozicionálásakor az objektum koordinátaegysége olvasható ki vagy állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a koordinátaegységet meghatározó **Integer** érték.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha az *érték*:	A koordinátaegység:
0	Azt jelzi, hogy a **ScaleHeight**, **ScaleWidth**, **ScaleLeft** vagy **ScaleTop** tulajdonságok közül egy vagy több beállítása saját egységet határoz meg.
1	Twip — 567 twip = 1 logikai cm. (Alapértelmezett.)
2	Pont — 28 pont = 1 logikai cm.
3	Képpont — a képernyő vagy nyomtató legkisebb felbontási eleme.
4	Karakter — az X tengelyen = 120 twip, az Y tengelyen 240 twip.
5	Inch — 1 inch = 2,54 cm.
6	Milliméter
7	Centiméter

A **ScaleHeight**, **ScaleWidth**, **ScaleLeft** és **ScaleTop** tulajdonságok együttesen a felhasználói igényeknek megfelelő, pozitív és negatív koordinátákkal rendelkező koordinátarendszert határoznak meg, és a **ScaleMode** tulajdonsággal a következőképpen hatnak egymásra:

- Bármely másik **Scale** tulajdonságot bármely értékre állítva a **ScaleMode** tulajdonság értéke 0 lesz.
- A **ScaleMode** értékét bármely nullánál nagyobb értékre állítva a **ScaleHeight** és **ScaleWidth** értéke az új mértékegység, a **ScaleLeft** és **ScaleTop** értéke pedig 0 lesz. A **CurrentX** és **CurrentY** tulajdonságok értékei az aktuális pont új koordinátáinak megfelelően módosulnak.

ScaleTop

Szintaxis: *objektum*.**ScaleTop** [= *érték*]

Ez a tulajdonság **Form** és **Printer** objektumon, továbbá **PictureBox** vezérlőelemen érvényes, és használatával a grafikus metódusok használatakor vagy egy vezérlőelem pozicionálásakor az objektum felső élének függőleges koordinátája olvasható ki vagy ál-

lítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a függőleges koordináta értékét meghatározó numerikus kifejezés.

A tulajdonságra vonatkozó egyéb tudnivalók értelemszerűen azonosak a **ScaleLeft** tulajdonságnál leírtakkal.

ScaleWidth

Szintaxis: *objektum*.**ScaleWidth** [= *érték*]

Ez a tulajdonság **Form**, **MDIForm** és **Printer** objektumon, továbbá **PictureBox** vezérlőelemen érvényes, és használatával a grafikus metódusok használatakor vagy egy vezérlőelem pozicionálásakor az objektumbelső vízszintes egységeinek száma olvasható ki vagy állítható be. Az **MDIForm** objektum esetén a tulajdonság a programfejlesztési fázisban nem elérhető, futásidőben pedig csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *érték* argumentum a vízszintes méretet meghatározó numerikus kifejezés. Ha a kifejezés negatív értékű, a koordinátarendszer a balról jobbra növekvő helyett jobbról balra növekvő irányítottságú lesz.

A tulajdonságra vonatkozó egyéb tudnivalók értelemszerűen azonosak a **ScaleHeight** tulajdonságnál leírtakkal.

ScrollBars

Szintaxis: *objektum*.**ScrollBars**

Ez a futásidőben csak olvasható tulajdonság **Column** és **MDIForm** objektumon, továbbá **Grid** és **TextBox** vezérlőelemen érvényes, és az objektumon függőleges vagy vízszintes gördítősáv meglétét meghatározó értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az **MDIForm** objektumon a tulajdonság által visszaadott érték és értelmezése a következő lehet:

True	Az űrlapnak függőleges, vízszintes vagy mindkét gördítősávja van. (Alapértelmezett.)
False	Az űrlapnak nincs gördítősávja.

A **Grid** vagy **TextBox** vezérlőelemen a tulajdonság által visszaadott érték és értel-

mezése a következő lehet:

Ha a **ScrollBars**:	Értelmezése:
0	*None* — A vezérlőelemnek nincs gördítősávja. (Alapértelmezett.)
1	*Horizontal* — A vezérlőelemnek vízszintes gördítősávja van.
2	*Vertical* — A vezérlőelemnek függőleges gördítősávja van.
3	*Both* — A vezérlőelemnek függőleges és vízszintes gördítősávja is van.

A **TextBox** vezérlőelemen az 1 (*Horizontal*), 2 (*Vertical*) vagy 3 (*Both*) beállításnál a **MultiLine** tulajdonságot **True** értékre kell állítani.

Bár a program futása alatt a Windows automatikusan a mutatóbillentyűk (FEL, LE, JOBB, BAL), valamint HOME, END stb. használatát is támogatja, a **ScrollBars** tulajdonság megkönnyíti a hosszabb szövegekben történő keresési műveleteket.

A gördítősávok csak akkor jelennek meg egy objektumon, ha annak tartalma túllépné az egyszerre megjeleníthető mértéket, például nem tudná megjeleníteni a szövegbeviteli mező vagy rács teljes tartalmát.

Ha a **ScrollBars** tulajdonság értéke **False**, az objektum a tartalmától függetlenül nem tartalmaz gördítősávot.

SelBookmarks

Szintaxis: *objektum*.**SelBookmarks**

Ez a tulajdonság **DBGrid** objektumon érvényes, és az objektumban kiválasztott összes rekord könyvjelzőinek halmazát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Amikor egy **DBGrid** objektumban egy rekordot kiválasztunk, könyvjelzői a **SelBookmark** által visszaadott halmazhoz adódnak.

SelCount

Szintaxis: *objektum*.**SelCount**

Ez a tulajdonság **ListBox** vezérlőelemen érvényes, és a vezérlőelemben kiválasztott elemek számát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Ha nincs kiválasztott elem, a **SelCount** által visszaadott érték 0, egyébként a kiválasztott elemek számát kapjuk vissza. A tulajdonság elsősorban a többszörös kiválasztási lehetőség esetén hasznos a kiválasztott elemek számának meghatározására.

Selected

Szintaxis: *objektum*.**Selected**(*index*) [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **FileListBox** és **ListBox** vezérlőelemen érvényes, és használatával a vezérlőelem egy elemének kiválasztottsági állapota olvasható ki vagy állítható be. A tulajdonság a **List** tulajdonság értékével azonos számú **Boolean** értékekből álló tömb. A tulajdonság szintaxisa a következő objektum-azonosítót és argumentumokat tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- Az *index* argumentum a vezérlőelem meghatározott elemének száma.
- A tulajdonság beállításához használható *érték* argumentum a meghatározott elem kiválasztottságát meghatározó **Boolean** típusú szám.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True Az elem kiválasztott állapotú.
False Az elem nem kiválasztott állapotú. (Alapértelmezett.)

A tulajdonság különösen egy listából több elem kiválasztásakor lényeges, hiszen a **ListIndex** csak a fókuszban lévő elem indexét adja meg, függetlenül attól, hogy az kiválasztott-e vagy sem, míg a **Selected** használatával az vizsgálható, hogy mely listaelemek kiválasztása történt meg. A tulajdonsággal a programkódból is kiválasztottá tehetők a kívánt listaelemek.

Ha csak egyetlen elem kiválasztott állapotú, a **ListIndex** tulajdonsággal meghatározható annak indexe, azonban több elem kiválasztottsága esetén a **ListIndex** a fókusz keretében lévő elem indexét adja vissza, függetlenül attól, hogy az éppen kiválasztott-e vagy sem.

SelectedItem

Szintaxis: *objektum*.**SelectedItem**

Ez a tulajdonság **DBComboBox** vezérlőelemen érvényes, és a vezérlőelemben kiválasztott rekord könyvjelzőjét meghatározó értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Amikor a **DBComboBox** vezérlőelem listamezőjében egy elem kiválasztott lesz, a **SelectedItem** tulajdonságban kapott könyvjelzőt a kiválasztott rekordnak a **Recordset** objektumban a **RowSource** tulajdonságnak megfelelő újrapozícionálásához lehet használni.

SelEndCol

Szintaxis: *objektum*.**SelEndCol** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **DBGrid** és **Grid** vezérlőelemen érvényes, és használatával a cellatartomány jobb oldali utolsó kiválasztott oszlopa olvasható ki vagy állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az utolsó oszlopot meghatározó numerikus kifejezés.

A tulajdonságot a vezérlőelem egy adott tartományának programkódból történő kiválasztására vagy a felhasználó által kiválasztott terület méretének kiolvasására lehet felhasználni.

A **SelEndCol** és a **SelEndRow** tulajdonságok együttesen a kiválasztott tartomány jobb alsó celláját jelölik ki.

Egy cella az aktuális kiválasztás elmozgatása nélkül a **Col** és **Row** tulajdonságokkal jelölhető ki.

SelEndRow

Szintaxis: *objektum*.**SelEndRow** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Grid** vezérlőelemen érvényes, és használatával a cellatartomány utolsó kiválasztott sora olvasható ki vagy állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az utolsó sort meghatározó numerikus kifejezés.

A tulajdonság használata értelemszerűen a **SelStartCol** tulajdonságnál leírtakkal azonos.

SelLength

Szintaxis: *objektum*.**SelLength** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **ComboBox**, **DBComboBox** és **TextBox** vezérlőelemen érvényes, és használatával a kiválasztott karakterek száma olvasható ki vagy állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a kiválasztott karakterek számát meghatározó numerikus kifejezés. Érvényes tartománya 0 és a vezérlőelem szövegbeviteli mezőjében lévő karakterek száma közötti érték. A negatív *érték* futásidejű hibát okoz.

A tulajdonságot beszúrási pont vagy tartomány kijelöléséhez, egy vezérlőelembeni karakterlánc-rész meghatározásához vagy szövegtörléshez, a **Clipboard** objektummal együtt pedig másolási, kivágási és beillesztési műveletek végrehajtására használható.

A **SelStart** és **SelText** tulajdonságok megváltozása esetén a **SelLength** értéke 0 lesz.

SelStart

Szintaxis: *objektum*.**SelStart** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **ComboBox**, **DBComboBox** és **TextBox** vezérlőelemen érvényes, és használatával szöveg esetén a kiválasztott szöveg kezdőpontja, egyébként pedig a beillesztési pont pozíciója olvasható ki vagy állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a kiválasztott szöveg kezdőpontját meghatározó numerikus kifejezés. Érvényes tartománya 0 és a vezérlőelem szövegbeviteli mezőjében lévő karakterek száma közötti érték. A szöveg hosszúságánál nagyobb *érték* a kezdőpontot a meglévő szöveg végére állítja be.

A **SelStart** megváltozása módosítja a beillesztési pont helyét, és ugyanakkor a **SelLength** értékét 0-ra állítja.

A tulajdonság használatával kapcsolatos további tudnivalók értelemszerűen azonosak a **SelLength** tulajdonságnál leírtakkal.

SelStartCol

Szintaxis: *objektum*. **SelStartCol** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **DBGrid** és **Grid** vezérlőelemen érvényes, és használatával a cellatartomány bal oldali első kiválasztott oszlopa olvasható ki vagy állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

- A tulajdonság beállításához használható *érték* argumentum az első oszlopot meghatározó numerikus kifejezés.

A tulajdonságot a vezérlőelem egy adott tartományának programkódból történő kiválasztására vagy a felhasználó által kiválasztott terület méretének kiolvasására lehet felhasználni.

A **SelStartCol** és a **SelStartRow** tulajdonságok együttesen a kiválasztott tartomány bal felső celláját jelölik ki.

Egy cella az aktuális kiválasztás elmozgatása nélkül a **Col** és **Row** tulajdonságokkal jelölhető ki.

SelStartRow

Szintaxis: *objektum*.**SelStartRow** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **DBGrid** és **Grid** vezérlőelemen érvényes, és használatával a cellatartomány első kiválasztott sora olvasható ki vagy állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az első sort meghatározó numerikus kifejezés.

A tulajdonság használata értelemszerűen a **SelStartRow** tulajdonságnál leírtakkal azonos.

SelText

Szintaxis: *objektum*.**SelText** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **ComboBox**, **DBComboBox** és **TextBox** vezérlőelemen érvényes, és használatával az aktuálisan kiválasztott szöveg olvasható ki vagy állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a kiválasztott szöveget tartalmazó karakteres kifejezés. Ha ez üres karakterlánc (""), nincs kiválasztott tartomány.

A **SelText** módosítása a kiválasztott szöveget az új karakterláncra cseréli és a **SelLength** értékét 0-ra állítja.

A tulajdonság használatával kapcsolatos további tudnivalók értelemszerűen azonosak a **SelLength** tulajdonságnál leírtakkal.

Shape

Szintaxis: *objektum*.**Shape** [= *érték*]

Ez a tulajdonság **Shape** vezérlőelemen érvényes, és használatával a vezérlőelem megjelenését meghatározó érték olvasható ki vagy állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a vezérlőelem megjelenését meghatározó **Integer** érték.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

Ha az *érték*:	Értelmezése:
0	*Rectangle* — Téglalap. (Alapértelmezett.)
1	*Square* — Négyzet
2	*Oval* — Ellipszis
3	*Circle* — Kör
4	*Rounded Rectangle* — Lekerekített sarkú téglalap
5	*Rounded Square* — Lekerekített sarkú négyzet

Shortcut

Szintaxis:

Ez a futásidőben nem elérhető tulajdonság **Menu** objektumon érvényes, és használatával a menüparancs gyorsabb kiadását segítő billentyűparancs a Menu Editor párbeszédpanelen állítható be. A használható billentyűkombinációkat a Menu Editor párbeszédpanel Shortcut listája tartalmazza.

A gyorsabb parancskiadásra a parancsok, menüparancsok és vezérlőelemek esetén a **Caption** tulajdonságukban használt "&" karakterrel kijelölt billentyűk is alkalmasak.

ShowInTaskbar

Szintaxis: *objektum*.**ShowInTaskbar**

Ez a futásidőben csak olvasható tulajdonság **Form** objektumon érvényes, és az objektumnak a Windows 95 feladattárban történő megjelenését meghatározó értéket ad vissza. Nem Windows 95 operációs rendszerben a tulajdonság hatástalan. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **ShowInTaskbar** lehetséges értékei és értelmezésük a következő:

True	A **Form** objektum megjelenik a feladattárban. (Alapértelmezett.)
False	A **Form** objektum nem jelenik meg a feladattárban.

A tulajdonságot arra lehet használni, hogy letiltsuk az alkalmazásunk párbeszédpaneljeinek a feladattárban történő megjelenését.

A **ShowInTaskbar** tulajdonság alapértelmezett értéke feltételezi a **Form** objektum **BorderStyle** tulajdonságának alapértelmezett (*Sizable*) beállítását, így a **BorderStyle** értékének módosítása a **ShowInTaskbar** tulajdonságot is módosíthatja.

Size

Szintaxis: *objektum*.**Shape** [= *érték*]

Ez a tulajdonság a **Field** és **Font** objektumon eltérő értelmezéssel használatos.

A **Field** objektumon a tulajdonság használatával egy szöveget tartalmazó **Field** objektum byte-ban mért maximális méretét, illetve szöveget vagy numerikus értéket tartalmazó **Field** objektum rögzített méretét meghatározó érték olvasható ki vagy állítható be. A tulajdonság **Fields** objektumgyűjteményhez még nem tartozó új **Field** objektum esetén írható és olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a maximális méretet meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentum lehetséges értékei a **Field** objektum **Type** tulajdonságától függőek és a következők lehetnek:

Ha a szimbolikus konstans:	*érték*:	**Type**:
dbBoolean	1	**True/False**
dbByte	1	Byte
dbInteger	2	**Integer**
dbLong	4	**Long**
dbCurrency	8	**Currency**
dbSingle	4	**Single**
dbDouble	8	**Double**
dbDate	8	Dátum/Idő
dbText	1—255	Szöveg
dbLongBinary	0	Long Binary (OLE objektum)
dbMemo	0	Memo

A konstansokat a DAO (Data Access Object) objektumtár tartalmazza.

Field objektumnál a **Size** tulajdonság használata attól a **Fields** halmazt tartalmazó objektumtól függ, amelyhez a **Field** tartozik:

Ha az objektum:	A tulajdonság használata:
Index	Nem támogatott.
QueryDef	Csak olvasható.
Recordset	Csak olvasható.
Relation	Nem támogatott.
TableDef	Csak olvasható.

Nem **Text** adattípussal létrehozott **Field** objektum esetén a **Type** tulajdonság beállítása automatikusan meghatározza a **Size** tulajdonság beállítását is, azzal már nem kell törődnünk. Az OLE objektum és Memo **Field** objektumoknál a **Size** beállítása mindig 0. A **Text** típusú **Field** objektumoknál a **Size** értéke legfeljebb 255 értékű egész szám lehet, amelynél a **Field** mérete a maximális lesz.

Az OLE objektum és Memo **Field** objektumoknál egy adott rekord adatmérete az objektum **FieldSize** metódusával határozható meg. Az OLE objektum és Memo mező méretét csak a rendszerünk erőforrásai vagy az adatbázis maximális mérete korlátozza.

Ugyanezen **Size** tulajdonság a **Font** objektumon az objektumban használt karakterméret kiolvasására vagy beállítására használható. Ebben az esetben a tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a használt karakterméretet pont egységben meghatározó, maximálisan 2048 **Integer** típusú érték.

A tulajdonságot a szöveg karakterméretének beállítására tudjuk használni. Az alapértelmezett méretet az operációs környezet határozza meg.

Mivel a **Font** objektum a programfejlesztési fázisban közvetlenül nem érhető el, így **Size** tulajdonságát a kívánt vezérlőelem Properties ablakban a Font tulajdonság kiválasztásával és a Properties gombra kattintással, majd a Font párbeszédpanel Size mezőjében a kívánt méret kiválasztásával állíthatjuk be. A **Size** tulajdonság futásidőben történő beállítása közvetlenül, a **Font** objektum számára a szükséges értéket megadva történhet.

SizeMode

Szintaxis: *objektum*.**SizeMode** [= *érték*]

Ez a tulajdonság **OLE** befogadó vezérlőelemen érvényes, és a vezérlőelem méretbeállítását vagy egy objektumának megjelenítési módját meghatározó érték olvasható ki vagy állítható be. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a méretezését vagy

megjelenítését meghatározó szimbolikus konstans vagy **Integer** érték.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	Értelmezése:
vbOLESizeClip	0	*Clip* — Az objektum az aktuális méretben jelenik meg. Ha az objektum nagyobb az **OLE** befogadó vezérlőelemnél, a vezérlőelem határvonalán túlnyúló része levágásra kerül. (Alapértelmezett.)
vbOLESizeStrech	1	*Strech* — Az objektum az **OLE** befogadó vezérlőelemet kitöltő méretet vesz fel. A kép nem tartja meg az objektum eredeti arányait.
vbOLESizeAutoSize	2	*Autosize* — Az **OLE** befogadó vezérlőelem a teljes objektum megjelenítéséhez szükséges méretet veszi fel.
vbOLESizeZoom	3	*Zoom* — Az objektum az **OLE** befogadó vezérlőelemet annyira kitöltő méretűvé válik, amely az eredeti arányainak megtartása mellett még lehetséges.

A **SizeMode** tulajdonság 2 (*Autosize*) értéke esetén az **OLE** befogadó vezérlőelem mérete automatikusan követi az objektum megjelenítési méretének változását, az átméretezést megelőzően azonban egy **Resize** esemény következik be. A **Resize** esemény *magasság* és *szélesség* argumentumai az objektumot létrehozó alkalmazás által meghatározott méretű objektum megjelenítésének optimális méretét adják meg. A vezérlőelem méretét a **Resize** eseményeljárásban a *magasság* és *szélesség* argumentumok értékeinek módosításával lehet beállítani.

SmallChange

Szintaxis: *objektum*.**SmallChange** [= *érték*]

Ez a tulajdonság a **HScrollBar** és **VScrollBar** vezérlőelemeken érvényes, és a vezérlőelem **Value** tulajdonságának a felhasználó által a gördítőnyilakra kattintáskor bekövetkező értékváltozás nagyságának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **Value** tulajdonság értékváltozását meghatározó, 1 és 32767 közötti (alapértelmezetten 1) **Integer** típusú szám.

A tulajdonságra vonatkozó tudnivalók azonosak a **LargeChange** tulajdonságnál leírtakkal.

Sort

Szintaxis: *objektum*.**Sort** [= *érték*]

Ez a tulajdonság eredményhalmaz és tábla típusú **Recordset** (valamint az elavult **Dynaset** és **Table**) objektumon érvényes, és az objektumban a rekordok rendezési módjának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy SQL utasítás **ORDER BY** záradékát az **ORDER BY** kulcsszó nélkül tartalmazó **String** típusú kifejezés.

A tulajdonságot egy objektumhoz beállítva a rendezés a következő **Recordset** objektumból történő létrehozásakor történik. A **Sort** tulajdonság beállítása felülírja egy **QueryDef** számára megadott bármely rendezést módot. (Az alapértelmezett rendezés mód az A—Z vagy 0—100 sorrend szerinti.)

Sok esetben gyorsabb egy új **Recordset** megnyitása a rendezési feltételt tartalmazó SQL utasítás használatával.

A tábla típusú **Recordset** objektumon a **Sort** tulajdonság nem érvényes, helyette az **Index** tulajdonságot kell használni.

Sorted

Szintaxis: *objektum*.**Sorted**

Ez a tulajdonság **ComboBox** és **ListBox** vezérlőelemen érvényes, és a vezérlőelem elemeinek betűrendbe állítottságát jelző értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Sorted** lehetséges értékei és értelmezésük a következő:

True	Az elemek a kis- és nagybetűs alakokat megkülönböztetve, betűrendbe állítva jelennek meg.
False	Az elemek nem betűrendbe rendezettek. (Alapértelmezett.)

A tulajdonság **True** értéke esetén a Visual Basic majdnem mindegyik szövegfeldolgozási műveletet — beleértve az elemek hozzáadása vagy eltávolítása során az indexek megváltoztatását is — ugyanilyen betűrendben végezi el.

Az **AddItem** metódus használatával az új elem — függetlenül attól, hogy a betűrendhez illeszkedik-e vagy sem — a lista meghatározott pozíciójába kerül, ezzel sértve a rendezettséget.

Source

Szintaxis 1: *objektum*.**Sorted**

Szintaxis 2: *objektum*.**Sorted** [= *érték*]

Ez a tulajdonság az **Error** és az **Err** objektumokon eltérő szintaxissal és értelmezéssel érvényes.

Az **Error** objektumon a **Source** tulajdonság a Szintaxis 1 szerint használatos, és a hibát eredetileg okozó objektum vagy alkalmazás nevét karakteres kifejezésként adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Source** tulajdonságban visszakapott karakteres kifejezés rendszerint az objektum osztályneve vagy programbeli azonosítója. Mivel a **Source** értéke mindig tartalmazza a hibát okozó objektum nevét, a kódunk az objektum leírásának megfelelően a hiba lekezelését megkísérelheti. Ha ez sikertelen, az Error objektum információit átadhatjuk a felhasználó számára, tájékoztatva a hibát okozó objektumról, a hiba okáról, jellegéről stb.

Az **On Error Resume Next** utasítás használata az **On Error GoTo** utasításnál kedvezőbb, mivel az objektumokhoz történő hozzáférés során bekövetkező hibák esetén lehetővé teszi a hibaszámot visszaadó objektum egyértelmű azonosítását. Minden egyes objektumkapcsolat után az **Error** objektum vizsgálatával azonnal meghatározható, hogy melyik objektumhoz történő hozzáférés során következett be hiba, melyik objektum állította be a hibaszámot az **Error.Number** tulajdonságba és eredetileg melyik okozta az **Error.Source** által meghatározott hibát.

Az **Err** objektumon a **Source** tulajdonság a Szintaxis 2 szerint használatos, és a hibát eredetileg okozó objektum vagy alkalmazás nevének kiolvasására vagy kiolvasására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a hibát okozó alkalmazás nevét meghatározó karakteres kifejezés.

A hiba programkódból történő létrehozása esetén a **Source** az alkalmazásunk programbeli azonosítója. Az OLE Automatizmus osztálymodulnál a **Source** a *projekt.osztály* kifejezésből származó nevet tartalmazza.

A **Source** tulajdonságra vonatkozó tudnivalók értelemszerűen azonosak az **Error** objektumnál már leírtakkal.

SourceDoc

Szintaxis:

Ez a tulajdonság **OLE** befogadó vezérlőelemen érvényes, és az objektum létrehozásakor használt fájlnév kiolvasására vagy beállítására használható, azonban a Visual Basic 4.0-s verziójában csak kompatibilitási okból maradt meg. Helyette a **CreateEmbed** és **CreateLink** metódusokat kell használni.

SourceField

Szintaxis: *objektum*.**SourceField**

Ez a programfejlesztési fázisban nem hozzáférhető, futásidőben csak olvasható tulajdonság **Field** objektumon érvényes, és használatával egy objektum eredeti adatforrásaként szereplő mező neve olvasható ki. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **SourceField** tulajdonság használhatóságát a **Field** objektum **Fields** halmazát tartalmazó objektum a következők szerint határozza meg:

Ha az objektum:	A tulajdonság használata:
Index	Nem támogatott.
QueryDef	Csak olvasható.
Recordset	Csak olvasható.
Relation	Nem támogatott.
TableDef	Csak olvasható.

A **SourceField** tulajdonság egy **Field** objektummal kapcsolatos eredeti mező- és táblaneveket adja meg.

SourceItem

Szintaxis: *objektum*.**SourceItem** [= *érték*]

Ez a tulajdonság **OLE** befogadó vezérlőelemen érvényes, és egy csatolt objektum létrehozásakor a fájlon belül csatolt adat kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a csatolt adatot meghatározó karakteres kifejezés.

A tulajdonság használata csak **OLETypeAllowed** = 0 (*Linked*) vagy **OLETypeAllowed** = 2 (*Either*) beállításnál megengedett. A csatolandó fájl kijelölésére a **SourceDoc** tulajdonságot kell használni.

Minden objektum saját szintaxist használ az adategységek leírására, így a **SourceItem** beállításához az objektum által felismerhető, az adott felhasználással összhangban álló adategység specifikálható, például az Excelben R1C1 vagy R12C5:R7C14 alakban.

A szintaxis meghatározásához a csatolt objektumnak a programfejlesztési fázisban a Paste Special paranccsal (az **OLE** befogadó vezérlőelemre a JOBB gombbal kattintva) történő létrehozását követően a Properties ablakban a SourceDoc tulajdonságot kiválasztva, a Settings mezőben találjuk meg a csatolt objektum karakterláncát. A legtöbb objektumnál a karakterláncban a csatolt fájl elérési útja és egy elválasztó karakter ("!" vagy "\") után találjuk a csatolt adat keresett szintaxisát.

A csatolt objektumok létrehozásakor a **SourceItem** a **SourceDoc** tulajdonsághoz láncolódik. Futásidőben a **SourceItem** egy üres ("") karakterláncot, míg a **SourceDoc** tulajdonság a csatolt fájl teljes elérési útját, egy elválasztó karaktert ("!" vagy "\") és az azt követő **SourceItem** értéket adja vissza.

SourceTable

Szintaxis: *objektum*.**SourceTable**

Ez a programfejlesztési fázisban nem hozzáférhető, futásidőben csak olvasható tulajdonság **Field** objektumon érvényes, és egy objektum eredeti adatforrásaként szereplő tábla neve olvasható ki. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **SourceTable** tulajdonsággal kapcsolatos tudnivalók értelemszerűen azonosak a **SourceField** tulajdonságnál leírtakkal.

SourceTableName

Szintaxis: *objektum*.**SourceTableName** [= *érték*]

Ez a tulajdonság **TableDef** objektumon érvényes, és egy csatolt tábla nevének vagy a lekérdezés alapjául szolgáló alaptábla eredeti adatforrásban szereplő nevének kiolvasására vagy beállítására használható. A **SourceTableName** tulajdonság egy alaptáblán csak olvasható, míg csatolt táblán vagy egy halmazhoz még nem csatolt objektumon írható és olvasható is. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a tábla nevét meghatározó **String** típusú kifejezés. Egy alaptáblánál a beállítás üres ("") karakterlánc.

SQL

Szintaxis: *objektum*.**SQL** [= *érték*]

Ez a tulajdonság **QueryDef** objektumon érvényes, és az objektum által végrehajtott SQL utasítás kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy SQL utasítást tartalmazó **String** típusú kifejezés.

Az **SQL** tulajdonság a lekérdezés végrehajtása során a rekordok kiválasztási, csoportosítási vagy rendezési módját meghatározó SQL utasítást tartalmazza. A lekérdezést eredményhalmaz és pillanatfelvétel típusú **Recordset** (valamint az elavult **Dynaset** és **Snapshot**) objektumbeli rekordok kiválasztására lehet használni. A lekérdezéssel rekordok visszaadása nélkül az adatok is módosíthatók.

Az **ORDER BY** vagy **WHERE** záradékkal a **Recordset** objektum rendezése vagy szűrése a **Sort** vagy **Filter** tulajdonság használatánál hatékonyabban végezhető.

A lekérdezésben használt SQL szintaxisnak összhangban kell állnia a Microsoft Jet adatbázis-szerkesztő által előírtakkal, kivéve azt az esetet, ha SQL utasításátadó lekérdezést használunk.

Az ODBC adatbázisokon egy SQL utasításátadó művelet végrehajtásának javasolt módja a **QueryDef** objektum használata. Az objektum **Connect** tulajdonságát egy ODBC adatbázisra beállítva egy külső kiszolgálónak átadásra kerülő, nem Jet adatbázis SQL adható meg.

Ha az SQL utasítás a lekérdezéshez paramétereket tartalmaz, azokat a lekérdezés végrehajtása előtt kell beállítani. Az újbóli beállításig ugyanazon paraméterértékek maradnak érvényesek minden egyes lekérdezés végrehajtása alkalmával.

StartMode

Szintaxis: *objektum*.**StartMode**

Ez a futásidőben csak olvasható tulajdonság **App** objektumon érvényes, és használatával az alkalmazás egyedülálló projektként vagy OLE Automatizálási OLE kiszolgálókénti indítását meghatározó érték olvasható ki. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **StartMode** tulajdonság lehetséges értékei a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	Értelmezése:
vbSModeStandalone	0	Az alkalmazás egyedülálló projektként indul. (Alapértelmezett.)

vbSModeAutomation	1	Az alkalmazás OLE Automatizálási OLE kiszolgálóként indul.

Ezek a konstansok a Visual Basic objektumtárban az Object Browser által adott listában találhatók.

Ha a programfejlesztés fázisában egy alkalmazás futását úgy kívánjuk követni, mintha az egy OLE Automatizálási OLE kiszolgálóként indult volna el, a Project Options párbeszédpanelen a **StartMode** = 1 értéket (**vbSModeAutomation**) kell beállítani, azonban a projekt aktuális **StartMode** értéke nem a futtatható (.EXE) fájl létrehozásakor ezen a párbeszédpanelen beállított értéktől, hanem a futásidőben történő indulásának módjától fog függeni.

Ha a **StartMode** = 1 értéke esetén nincs a projektben publikus osztály, az alkalmazás befejezésére az **End** utasítást, a Run menü End parancsát vagy az eszköztárat kell használni. Amennyiben a Control menü Close parancsát használjuk, az űrlap ugyan bezáródik, de a projekt továbbra is futó állapotú marad.

Stretch

Szintaxis: *objektum*.**Stretch** [= *érték*]

Ez a tulajdonság **Image** vezérlőelemen érvényes, és a grafikának a vezérlőelem méretéhez illeszkedő megjelenését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a grafika újraméretezését meghatározó **Boolean** típusú kifejezés.

A **Stretch** lehetséges értékei és értelmezésük a következő:

True	A grafika a vezérlőelem méretéhez alkalmazkodóan és annak méretváltozását automatikusan követve méreteződik.
False	A vezérlőelem a grafika méretéhez alkalmazkodóan és annak méretváltozását automatikusan követve méreteződik. (Alapértelmezett.)

Ha a **Stretch** tulajdonság értéke **True**, a vezérlőelem méretének módosításakor az általa tartalmazott grafika is átméreteződik.

StrikeThrough

Szintaxis: *objektum*.**StrikeThrough** [= *érték*]

Ez a tulajdonság **Font** objektumon érvényes, és az objektum áthúzott karakterstílusú megjelenését meghatározó érték kiolvasására vagy beállítására használható. A tulajdon-

ság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a karakterstílust meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

True	Az áthúzott karakterstílus érvényes.
False	Az áthúzott karakterstílus nem érvényes. (Alapértelmezett.)

Mivel a **Font** objektum a programfejlesztési fázisban közvetlenül nem érhető el, **StrikeThrough** tulajdonságának értékét a kívánt vezérlőelem Properties ablakban a Font tulajdonság kiválasztásával és a Properties gombra kattintással, majd a Font párbeszédpanel Strikethrough választómező kiválasztásával állíthatjuk be. A tulajdonság futásidőben történő beállítása közvetlenül, a **Font** objektum számára a szükséges értéket megadva történhet.

Style

Szintaxis: *objektum*.**Style**

Ez a futásidőben csak olvasható tulajdonság **ComboBox** és **DBCombo** vezérlőelemen érvényes, és a kombinált mezőtípusát és listarészének működésmódját meghatározó érték olvasható ki. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Style** lehetséges értékei és értelmezésük a következő:

Ha a **Style**:	Értelmezése:
0	*Dropdown Combo* — A vezérlőelem szövegbeviteli mezőt és legördülő listát tartalmaz, amelyet a hozzá tartozó nyílra kattintva lehet megjeleníteni. A felhasználó a listából választhat vagy a szövegbeviteli mező be írhat. (Alapértelmezett.)
1	*Simple Combo* — A vezérlőelem szövegbeviteli mezőt és állandó listát tartalmaz. A felhasználó a listából választhat vagy a szövegbeviteli mezőt írhat. Alapértelmezetten a vezérlőelem olyan nagyságú, hogy a lista nem jelenik meg. A megjelenítendő lista hosszát a **Height** tulajdonság határozza meg.
2	*Dropdown List* — Csak egy listából lehet választani.

A **Style** tulajdonság megfelelő beállításához a következőket kell figyelembe venni:

- A **Style** = 0 (*Dropdown Combo*) vagy **Style** = 1 (*Simple Combo*) beállításnál a felhasználó egy listáról választhat vagy választását egy szövegbeviteli mező be írhatja be. A *Dropdown Combo* beállítás helytakarékosabb, mivel a lista a fel-

használó általi választást követően bezárul.

- A **Style** = 2 (*Dropdown list*) beállításnál egy rögzített lista jelenik meg, amelyből a felhasználó egy elemet választhat, majd ezután a listarész bezárul.

System

Szintaxis: *objektum*.**System**[= *érték*]

A **System** tulajdonság **FileListBox** vezérlőelemekre vonatkozik, és a vezérlőelemben listázandó fájlok **System** attribútumát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True	A **FileListBox** vezérlőelemben a **System** attribútumú fájlokat jeleníti meg. (Alapértelmezett.)
False	A **FileListBox** vezérlőelemben nem jeleníti meg a **System** attribútumú fájlokat.

A tulajdonsággal a **FileListBox** vezérlőelemben megjelenítendő, az operációs környezetben használt standard fájlattribútum állítható be. Ha a tulajdonság beállítása futásidőben, programkódból történik, a **FileListBox** vezérlőelem újra beállítódik és azon csak az adott attribútumú fájlok jelennek meg.

TabIndex

Szintaxis: *objektum*.**TabIndex** [= *index*]

Ez a tulajdonság **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemen érvényes, és az űrlapon belüli léptetési sorrend kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *index* argumentum egy 0 és a **TabIndex** tulajdonságú űrlapra érvényes (vezérlőelemszám -1) érték közötti egész szám. Ha az *index* negatív, hiba következik be.

Alapértelmezetten a Visual Basic a vezérlőelemek számára a létrehozásuk sorrend-

jének megfelelően állítja be a **TabIndex** tulajdonságukat. Minden egyes új vezérlőelem a léptetési sor végére kerül. Ha a vezérlőelem **TabIndex** tulajdonságát módosítjuk, a vezérlőelemek közül törlünk vagy új vezérlőelemet szúrunk be, a Visual Basic automatikusan a többi objektum **TabIndex** tulajdonságát is átszámozza.

A menük és időzítők kivételével minden vezérlőelem szerepel a léptetési rendben. Futásidőben a láthatatlan, letiltott és a fókuszt nem fogadó (**Frame** és **Label**) vezérlőelemek a léptetési sorban maradnak, azonban a Visual Basic a léptetés során ezeket átugorja.

A **TabIndex** tulajdonságra a **ZOrder** metódus hatástalan.

A vezérlőelem közvetlen elérését biztosító billentyűket a léptetési sorrend változása nem befolyásolja. Egy **Frame** vagy **Label** közvetlen elérését biztosító billentyű leütésekor a léptetési sorban következő, fogadóképes vezérlőelem kerül fókuszba.

Az ASCII szövegként mentett űrlapok betöltésekor az űrlap leírásában nem listázott **TabIndex** tulajdonsággal rendelkező vezérlőelemek automatikusan egy **TabIndex** értéket kapnak. Ha a következőként betöltött vezérlőelemek **TabIndex** értékei a már meglévő értékekkel ütköznének, a vezérlőelemek automatikusan új értéket kapnak.

Table

Szintaxis: *objektum*.**Table** [= *érték*]

Ez a tulajdonság **Relation** objektumon érvényes, és az objektum elsődleges táblájának nevét meghatározó érték kiolvasására vagy beállítására használható. A **Table** tulajdonság egy halmazhoz még nem csatolt új objektumon írható és olvasható, míg egy **Relations** halmazban lévő **Relation** objektumnál csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **TableDefs** halmazban lévő tábla nevét meghatározó **String** típusú kifejezés.

A **Table** és a **ForeignTable** tulajdonság együttes használatával egy Relation objektum határozható meg, amely két tábla vagy lekérdezés közötti kapcsolatot képvisel. A **Table** tulajdonság értékét az elsődleges tábla vagy a **QueryDef** objektum **Name** tulajdonságára, majd a **ForeignTable** tulajdonságot az idegen (hivatkozott) **TableDef** vagy **QueryDef** objektum **Name** tulajdonságára kell beállítani. A két objektum közötti viszonyt az **Attributes** tulajdonság adja meg.

TabStop

Szintaxis: *objektum*.**TabStop** [= *érték*]

Ez a tulajdonság **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBList**,

DirListBox, **DriveListBox**, **FileListBox**, **Grid**, **HScrollBar**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **TextBox** és **VScrollBar** vezérlőelemen érvényes, és a vezérlőelem TAB billentyűvel történő fókuszba állíthatóságát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az adott vezérlőelemet megállási pontként megadó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True	A vezérlőelem a TAB billentyűvel fókuszba hozható. (Alapértelmezett.)
False	A léptetés átugorja a vezérlőelemet, bár az megtartja az aktuális helyét a **TabIndex** által meghatározott sorban.

Ezzel a tulajdonsággal egy űrlapon a léptetési sorhoz egy elem hozzáadható vagy abból eltávolítható. A vezérlőelemek tömbjét egyetlen léptetési elemként lehet kezelni, ha a tömb első elemén a **TabStop** tulajdonság **True**, a többin pedig **False** értékű.

Tag

Szintaxis: *objektum*.**Tag** [= *érték*]

Ez a tulajdonság **Form**, **MDIForm** és **Menu** objektumon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **CommonDialog**, **DBCombo**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **Line**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox**, **Timer** és **VScrollBar** vezérlőelemen érvényes, és a program számára szükséges különleges adatokat tároló kifejezés kiolvasására vagy beállítására használható. Eltérően az egyéb tulajdonságoktól, a **Tag** tulajdonság értékét a Visual Basic nem használja, szerepe csak az űrlap vagy vezérlőelem azonosítására korlátozódik. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az adott vezérlőelemet azonosító karakteres kifejezés. Alapértelmezett értéke üres ("") karakterlánc.

Ezzel a tulajdonsággal az objektumhoz az egyéb tulajdonságainak befolyásolása és mellékhatások veszélye nélkül lehet egyedi azonosító karakterláncot kapcsolni. A **Tag** tulajdonság például egy eljárásnak változóként átadandó vezérlőelem vagy **MDIForm** objektum azonosságának ellenőrzéséhez használható.

Egy űrlap új példányához a létrehozásakor egy egyedi értékű **Tag** tulajdonságot kell beállítani.

TaskVisible

Szintaxis: *objektum*.**TaskVisible** [= *érték*]

Ez a tulajdonság **App** objektumon érvényes, és az alkalmazásnak a Windows feladatsorában való megjelenését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az alkalmazásnak a feladatsorban való megjelenését meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True Az alkalmazás megjelenik a Windows feladatsorában. (Alapértelmezett.)
False Az alkalmazás nem jelenik meg a Windows feladatsorában.

A **TaskVisible** tulajdonság csak **False** beállítású lehet az olyan alkalmazásokban, amelyek nem jelenítenek meg felhasználói felületet, például OLE kiszolgálók esetén, amelyek nem tartalmaznak vagy jelenítenek meg **Form** objektumot. Az alkalmazói felületet megjelenítő alkalmazások esetén a **TaskVisible** tulajdonság automatikusan **True** értékű lesz.

Text

Szintaxis: *objektum*.**Text** [= *érték*]

Ez a tulajdonság **Column** objektumon, továbbá **ComboBox**, **DBCombo**, **DBGrid**, **DBList**, **Grid**, **ListBox** és **TextBox** vezérlőelemen érvényes, és az objektum vagy vezérlőelem tartalmát a következőképpen határozza meg:

- A **Style** = 0 (*Dropdown Combo*) és **Style** = 1 (*Simple Combo*) tulajdonságú **ComboBox**, valamint a **TextBox** vezérlőelemeknél a szövegbeviteli mező tartalmának kiolvasására vagy beállítására használható.
- A **Style** = 2 (*Dropdown List*) tulajdonságú **ComboBox**, valamint a **ListBox** vezérlőelemeknél a tulajdonság a programfejlesztési fázisban csak olvasható, futásidőben pedig írható és olvasható, és a lista kiválasztott — a `List(ListIndex)` kifejezés által visszaadottal megegyező értékű — elemét adja vissza.
- A **Grid** vezérlőelemeknél a tulajdonság a programfejlesztési fázisban csak olvasható, és egy cella vagy cellatartomány tartalmának kiolvasására vagy beállítására használható.

A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a szöveget tartalmazó karakteres kifejezés. A kifejezés hosszúsága **TextBox** vezérlőelemnél maximum 2048 karakter, illetve a **MultiLine** tulajdonságának **True** értéknél közel 32K lesz.

A programfejlesztési fázisban a **Text** tulajdonság alapértelmezett értéke a következő:

- **ComboBox** és **TextBox** vezérlőelemeknél a vezérlőelem **Name** tulajdonsága.
- **ListBox** vezérlőelemeknél üres ("") karakterlánc.

A **Text** és a **FillStyle** tulajdonság használatával lehet szöveget a **Grid** vezérlőelem egy kijelölt cellatartományához hozzáadni. Amikor a **FillStyle** értéke 0, a **Text** tulajdonságban adott szöveg csak a **Row** és **Col** által kijelölt cellához adódik, míg a **FillStyle** = 1 beállítás mellett mindazokhoz a cellákhoz, amelyek **CellSelected** tulajdonsága **True**. Egy cellatartományt a **Clip** tulajdonság felhasználásával is fel lehet tölteni a **Clipboard** információblokkjának a **Grid** vezérlőelembe illesztésével.

Title

Szintaxis: *objektum*.**Title** [= *érték*]

Ez a tulajdonság **App** objektumon érvényes, és használatával az alkalmazásnak az MS Windows Task List listájában megjelenő címe olvasható ki vagy állítható be. Ha a tulajdonság értéke futásidőben megváltozik, a változás nem mentődik az alkalmazással együtt. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az alkalmazás címét tartalmazó, legfeljebb 40 karakterből álló kifejezés.

A tulajdonság a programfejlesztési fázisban a File menü Make EXE parancsával megnyitott párbeszédpanelen állítható be.

Top

Szintaxis: *objektum*.**Top** [= *érték*]

Ez a tulajdonság **Form**, **Column** és **MDIForm** objektumon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **CommonDialog**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE befogadó**, **OptionButton**, **PictureBox**, **Shape**, **TextBox**, **Timer** és

VScrollBar vezérlőelemen érvényes, és az adott objektum belső bal oldali éle és a befogadójának bal oldali éle közötti távolság kiolvasására vagy beállítására használható. A tulajdonság **Timer** vezérlőelemen futásidőben nem elérhető. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a távolságot megadó **Single** típusú numerikus kifejezés.

A **Left** tulajdonság használatra vonatkozó tudnivalók azonosak a **Left** tulajdonságnál leírtakkal.

ToPage

Szintaxis: *objektum*.**ToPage** [= *érték*]

Ez a **CommonDialog** (Print Dialog) vezérlőelemen használható tulajdonság a To szövegbeviteli mező értékének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az utolsó nyomtatandó oldal számát megadó numerikus kifejezés.

A **ToPage** tulajdonság csak a **Flags** tulajdonság **cflPageNums** jelzőjének beállított állapotában érvényes.

TopIndex

Szintaxis: *objektum*.**TopIndex** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **FileListBox** és **ListBox** vezérlőelemen érvényes, és a vezérlőelem listájának legfelső pozíciójában megjelenő elemet meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a lista legfelső pozíciójában megjelenő elem sorszáma. Az alapértelmezett *érték* 0, azaz a lista első eleme.

A tulajdonság a **FileListBox** vagy **ListBox** vezérlőelemen a listaelemek kiválasztása nélküli görgetésre használható.

Ha a **ListBox** vezérlőelem **Columns** tulajdonságának értéke 0, az elem akkor jelenik meg a legfelső pozícióban, ha elegendő további elem marad a lista alatta látható részé-

nek feltöltéséhez. Ha a **Columns** tulajdonság értéke nagyobb, mint 0, az elem oszlopa anélkül kerül a bal szélső pozícióba, hogy az elem oszlopon belüli helyzete változna.

TopRow

Szintaxis: *objektum*.**TopRow** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **DBGrid** és **Grid** vezérlőelemeken érvényes, és egy vezérlőelem legfelső megjelenített sorának kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a legfelső látható sor számát megadó numerikus kifejezés.

A **TopRow** tulajdonság programkódban történő használatával a **DBGrid** és **Grid** vezérlőelemek programozottan görgethetők. Ugyanezen vezérlőelemek bal oldali legszélső oszlopát a **LeftCol** tulajdonsággal lehet meghatározni.

A beállítható legnagyobb szám a **Grid** vezérlőelemben látható sorok száma -1 érték lehet, az ennél nagyobb szám megadása hibát okoz.

TrackDefault

Szintaxis: *objektum*.**TrackDefault** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Printer** objektumon érvényes, és az objektumnak mindig ugyanazon nyomtatóra vagy az operációs rendszer Control Panel párbeszédpaneljén alapértelmezett nyomtató megváltoztatásakor az új nyomtatóra történő mutatását meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az *objektum* által mutatott nyomtatót meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei és értelmezésük a következő:

True A **Printer** objektum másik nyomtatóra mutat, ha az operációs rendszer Control Panel párbeszédpaneljén az alapértelmezett nyomtató megváltozik. (Alapértelmezett.)

False A **Printer** objektum akkor is ugyanarra a nyomtatóra mutat, ha közben az operációs rendszer Control Panel párbeszédpaneljén az alapértelmezett nyomtató megváltozott.

A **TrackDefault** tulajdonság beállításának nyomtatási folyamat közbeni megváltoztatása egy implicit **EndPage** utasítást küld a **Printer** objektumra.

Transactions

Szintaxis: *objektum*.**Transactions**

Ez a tulajdonság **Dataset** és **Recordset** (valamint az elavult **Dynaset**, **Snapshot** és **Table**) objektumokon érvényes, és a későbbiekben törölhető vagy menthető módosítássorozat rögzítésének objektum általi támogatottságát jelző értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Transactions** lehetséges értékei és értelmezésük a következő:

Ha a **Transactions**:	Értelmezése:
True	Az objektum támogatja a tranzakciókat.
False	Az objektum nem támogatja a tranzakciókat. A pillanatfelvétel típusú **Recordset** (és az elavult **Snapshot**) objektumon csak ez az érték érvényes.

Ha egy eredményhalmaz vagy tábla a icrosoft Jet adatbázis-szerkesztő tábláján alapul, a **Transaction** tulajdonság **True** lesz, és a tranzakciók használhatók, de más adatbázisok nem biztosan támogatják a tranzakciós műveleteket. Például egy Paradox táblán alapuló eredményhalmaz típusú **Recordset** esetén nem lehet tranzakciókat használni.

A **BeginTrans** metódusnak egy **Recordset** objektum **Workspace** objektumán történő használata előtt a **Transactions** tulajdonság vizsgálatával kell meggyőződni arról, hogy a tranzakciók végrehajtása támogatott-e. A nem támogatott objektumokon a **BeginTrans CommitTrans** és **Rollback** metódusok hatástalanok lesznek.

TwipsPerPixelX

Szintaxis: *objektum*.**TwipsPerPixelX**

Ez a tulajdonság **Screen** és **Printer** objektumokon érvényes, és az objektum vízszintes irányban érvényes képpontméretét twip egységben adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Mivel a Windows API rutinok számára a méretet általában képpontban kell megadni, a **TwipsPerPixelX** tulajdonsággal a twip egységben adott értékek képpontra történő gyors konvertálása az objektum **ScaleMode** tulajdonságának módosítása nélkül lehetséges.

TwipsPerPixelY

Szintaxis: *objektum*.**TwipsPerPixelY**

Ez a tulajdonság **Screen** és **Printer** objektumokon érvényes, és az objektum függőleges irányban érvényes képpontméretét twip egységben adja vissza. A tulajdonságra vonatkozó tudnivalók azonosak a **TwipsPerPixelX** tulajdonságnál leírtakkal.

Type

Szintaxis 1: *objektum*.**Type** [= *érték*]

Szintaxis 2: *objektum*.**Type**

Ezt a tulajdonságot **Field**, **Parameter**, **Property**, **QueryDef** és **Recordset** objektumon, illetve **Picture** objektumon eltérő szintaxissal és értelmezéssel lehet használni.

A **Field**, **Parameter**, **Property**, **QueryDef** és **Recordset** objektumon a **Type** tulajdonság a Szintaxis 1 szerint érvényes, és az objektum adat vagy műveleti típusát meghatározó érték kiolvasására vagy beállítására használható. A **Type** tulajdonság a **Field** és **Property** esetén az objektum halmazhoz vagy másik objektumhoz csatolásáig írató és olvasható, után pedig csak olvasható, míg a **Property**, **QueryDef** és **Recordset** esetén csak olvasható lesz. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az adat vagy műveleti típust meghatározó szimbolikus konstans vagy **Integer** típusú állandó.

Az *érték* argumentumnak egy **Field**, **Parameter** vagy **Property** objektumon lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	Értéke:	Értelmezése:
dbBoolean	1	Igen/Nem
dbByte	2	**Byte**
dbInteger	3	**Integer**
dbLong	4	**Long**
dbCurrency	5	**Currency**
dbSingle	6	**Single**
dbDouble	7	**Double**
dbDate	8	Dátum/Időpont
dbText	10	Szöveg
dbLongBinary	11	Hosszú bináris (OLE objektum)
dbMemo	12	Memo

A hosszú bináris adattípus a **String** adattípussal megegyező, azonban helyette a

Variant adattípus használata javasolt, mivel az **Null** érték tárolását is lehetővé teszi.

Az *érték* argumentumnak egy **QueryDef** objektumon lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	A **QueryDef** típusa:
dbQSelect	Kiválasztási
dbQAction	Műveleti
dbQCrosstab	Keresztmezős
dbQDelete	Törlési
dbQUpdate	Frissítő
dbQAppend	Hozzáadó
dbQMakeTable	Táblakészítő
dbQDDL	Adat definiáló
dbQSQLPassThrough	Utasításátadó
dbQSetOperation	Egyesítő
dbQSPTBulk	A **dbQSQLPassThrough** konstanssal együtt rekordot nem visszaadó lekérdezést megadó

Az *érték* argumentumnak egy **Recordset** objektumon lehetséges értékei és értelmezésük a következő:

Ha a szimbolikus konstans:	A **Recordset** típusa:
dbOpenTable	Tábla
dbOpenDynaset	Eredményhalmaz
dbOpenSnapshot	Pillanatfelvétel

Ezek a konstansok a DataAccess (DAO) objektumtárban az Object Browser által adott listában találhatók.

Az objektum típusától függően a **Type** tulajdonság a következőket jelöli:

Ha az objektum:	A **Type** jelzése:
dbOpenTable	Az objektum adattípusa
dbOpenDynaset	A lekérdezés típusa
dbOpenSnapshot	A **Recordset** típusa

Új **Field**, **Parameter** vagy **Property** objektumnak egy **Index**, **QueryDef**, **Recordset** vagy **TableDef** objektumhoz csatolásakor hiba következik be, ha az alapjául szolgáló adatbázis nem támogatja az új objektum számára adott adattípust.

Egy SQL utasításátadó lekérdezés létrehozásához nem szabad a **dbSQLPassThrough** konstanst beállítani, mivel ezt a Microsoft Jet adatbázis-készítő akkor állítja be, amikor egy **QueryDef** objektumot hozunk létre és beállítjuk a **Connect** tulajdonságot.

A **Picture** objektumon a **Type** tulajdonság a Szintaxis 2 szerint érvényes, és az ob-

jektum grafikus formáját adja vissza. A **Type** programfejlesztési fázisban nem elérhető, futásidőben pedig csak olvasható. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Type** tulajdonság lehetséges értékei a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	Értelmezése:
vbPicTypeBitmap	1	Bitkép (.BMP fájl)
vbPicTypeMetafile	2	Metafájl (.WMF fájl)
vbPicTypeIcon	3	Ikon (.ICO fájl)

Ezek a konstansok a Visual Basic objektumtárban az Object Browser által adott listában találhatók.

UBound

Szintaxis: *objektum*.**UBound**

A tulajdonság vezérlőelemek tömbjén érvényes, és a tömbben használt legnagyobb sorszámot adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az **UBound** tulajdonság a tömbben utolsó elem **Index** tulajdonságának értékével azonos.

Underline

Szintaxis: *objektum*.**Underline** [= *érték*]

Ez a tulajdonság **Font** objektumon érvényes, és az objektum aláhúzott karakteres megjelenését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a karakterstílust meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

True Az aláhúzott karakterstílus érvényes.
False Az aláhúzott karakterstílus nem érvényes. (Alapértelmezett.)

Mivel a **Font** objektum a programfejlesztési fázisban közvetlenül nem érhető el, **Underline** tulajdonságának értékét a kívánt vezérlőelem Properties ablakban a Font tulajdonság kiválasztásával és a Properties gombra kattintással, majd a Font párbeszédpanel Underline választómező kiválasztásával állíthatjuk be. A tulajdonság futásidőben történő beállítása közvetlenül, a **Font** objektum számára a szükséges értéket

megadva történhet.

Unique

Szintaxis: *objektum*.**Unique** [= *érték*]

Ez a tulajdonság **Index** objektumon érvényes, és az objektumnak egy táblához tartozó egyedi (kulcs)index voltát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az egyediséget jelölő **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

True Az **Index** objektum egy egyedi indexet képvisel.
False Az **Index** objektum egy nem egyedi indexet képvisel.

Az elsődleges index egy vagy több mezőt tartalmaz, amely előre adott sorrendben és egyedileg azonosít minden rekordot egy táblában. Mivel az indexmező egyedi, az **Index** objektum **Unique** tulajdonsága **True** értékű. Ha az elsődleges index egynél több mezőt tartalmaz, az egyes mezők tartalmazhatnak kettős értékeket is, azonban az értékek minden egyes kombinációjának egyedinek kell lennie.

Ha egy **Index** objektum **Unique** és a **Primary** tulajdonságai egyaránt **True** értékűek, az index egyedi és elsődleges, és előre adott sorrendben és egyedileg azonosít minden rekordot egy táblában. Ha a **Primary** tulajdonsága **False**, az index másodlagos, és a rekordokat táblabeli azonosító szolgáltatása nélkül logikailag előre adott sorrendbe rendezi.

Updatable

Szintaxis: *objektum*.**Updatable**

Ez a tulajdonság **Database**, **QueryDef**, **Recordset** és **TableDef** (valamint az elavult **Dynaset**, **Snapshot** és **Table**) objektumokon érvényes, és az objektum frissíthetőségét jelző értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Az **Updatable** lehetséges értékei és értelmezésük a következő:

True Az objektum módosítható vagy frissíthető.
False Az objektumot nem lehet módosítani vagy frissíteni. A pillanatfelvétel típusú **Recordset** (és az elavult **Snapshot**) objektumon csak ez az érték érvényes.

Ha az **Updatable** tulajdonság értéke **True**, az objektumtól függően a következő hatása van:

- A **Database** objektum módosítható.
- A **QueryDef** objektumlekérdezési definíció módosítható.
- A **Recordset**, **Dynaset** vagy **Table** objektum rekordok frissíthetők.
- A **TableDef** objektum tábladefiníció módosítható.

Az **Updatable** tulajdonság az újonnan létrehozott **TableDef** objektumok esetén **True**, és csatolt **TableDef** objektumok esetén pedig **False** lesz. Az új **TableDef** objektumot csak olyan adatbázishoz lehet csatolni, amelynek aktuális felhasználója írási jogosultsággal rendelkezik.

Sok objektumtípus tartalmazhat nem frissíthető mezőket. Ezek a mezők lehetnek rögzítettek, automatikusan növekvő adatokat, például indexváltozókat tartalmazók, vagy a eredményhalmaz halmaz származhat frissíthető és nem frissíthető táblákat kombináló lekérdezésből.

A kizárólag nem frissíthető mezőket tartalmazó objektum **Updatable** tulajdonsága **False**, de a legalább egy frissíthető mezőt tartalmazó objektumé tulajdonsága **True** értékű lesz. Egy nem frissíthető mezőhöz új érték beállítását megkísérelve hiba következik be.

Ha a lekérdezés frissíthető, a **QueryDef** objektum **Updatable** tulajdonsága még akkor is **True**, ha az eredményül kapott **Recordset** nem frissíthető lesz.

Mivel egy frissíthető objektum nem frissíthető mezőket is tartalmazhat, egy **Recordset** objektum **Fields** halmazának vizsgálatával lehet egy rekordjának szerkesztése előtt az egyes mezők **DataUpdatable** tulajdonságát ellenőrizni.

UpdateOptions

Szintaxis: *objektum*.**UpdateOptions** [= *érték*]

Ez a tulajdonság **OLE** befogadó vezérlőelemen érvényes, és egy objektumnak a csatolt adat változásakor bekövetkező frissítési módját meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a frissítési módot meghatározó szimbolikus konstans vagy **Integer** típusú érték.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

Ha a szimbolikus konstans:	Értéke:	Értelmezése:
vbOLEAutomatic	0	*Automatic* — Az objektumot a csatolt adat minden megváltozására automatikusan frissíti. (Alapértelmezett.)
vbOLEFrozen	1	*Frozen* — Az objektum mindannyiszor frissítődik, amikor a felhasználó a csatolt dokumentumot az azt létrehozó alkalmazáson belül menti.
vbOLEManuel	2	*Manual* — Az objektum csak az **Update** metódussal frissíthető.

A tulajdonság olyan csatolt objektumok esetén hasznos, amelyeknél a csatolt adatot más felhasználó vagy alkalmazás is elérheti és módosíthatja.

Egy objektum adatának minden módosítása egy **Updated** eseményt okoz.

UseMnemonic

Szintaxis: *objektum*.**UseMnemonic** [= *érték*]

Ez a tulajdonság **Label** vezérlőelemen érvényes, és a vezérlőelem **Caption** tulajdonságában az amperstand (&) karakterrel történő hívóbetű-kijelölés lehetőségét meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **Label** vezérlőelem hívóbetű-kijelölés lehetőségét meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

True	A vezérlőelem **Caption** tulajdonságának karakterláncában bárhol megjelenő amperstand (&) a következő karaktert hívóbetűvé teszi. Az amperstand maga nem jelenik meg karakterként a **Label** vezérlőelemen. (Alapértelmezett.)
False	A vezérlőelem **Caption** tulajdonságának karakterláncában használt amperstand (&) a **Label** vezérlőelem szövegében "&" karakterként jelenik meg.

Futásidőben az ALT + a **Label** vezérlőelem szövegében megadott hívóbetű billentyűjének leütése azt a vezérlőelemet hozza a fókuszba, amely a **Label** vezérlőelemet követi a léptetési sorban.

UserName

Szintaxis: *objektum*.**UserName** [= *érték*]

Ez a tulajdonság **Container**, **Document** és **Workspace** objektumon érvényes, és vagy egy felhasználót vagy felhasználók csoportját, vagy egy **Workspace** objektum tu-

lajdonosát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság **Container** és **Document**, valamint egy halmazhoz még nem csatolt **Workspace** objektum esetén írható és olvasható, míg meglévő **Workspace** objektum esetén csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **Users** halmazban egy **User** objektum, vagy egy **Groups** halmazban egy **Group** objektum nevét tartalmazó **String** típusú kifejezés.

Az objektumtól függően a **UserName** tulajdonság a következő jelentésű:

- Egy **Container** és **Document** objektum hozzáférési jogosultságának manipulációja esetén egy felhasználó vagy felhasználócsoport.
- Egy **Workspace** objektum tulajdonosa. A **Container** és **Document** objektumok esetén a **UserName** és **Permission** tulajdonságok együttes használatával lehet írási vagy olvasási jogosultságokat megadni.

Egy adott felhasználó vagy felhasználócsoport jogosultságainak kiolvasásához vagy beállításához először a **UserName** tulajdonságot a kívánt felhasználó vagy felhasználócsoport nevére kell beállítani, majd kiolvasható vagy beállítható a jogosultságokat meghatározó **Permission** tulajdonság értéke.

Egy **Workspace** objektum esetén a **UserName** tulajdonság az objektum tulajdonosát határozza meg. A **Workspace** objektum számára a **Workspaces** objektumhalmazhoz csatolása előtt a **UserName** tulajdonság beállításával adhatunk tulajdonost.

V1xNullBehavior

Szintaxis: *objektum*.**V1xNullBehavior** [= *érték*]

Ez a tulajdonság **Database** objektumon érvényes, és a Text vagy Memo mezők feltöltésére a kódban használt üres ("") karakterlánc **Null** értékké konvertálását meghatározó érték kiolvasására vagy beállítására használható. A **V1xNullBehavior** a Microsoft Jet 1.x-s verzióval készített adatbázison csak akkor érvényes, ha az már a Jet 2.5-s vagy Jet 3.0-s verzióra konvertálódott. A Jet a tulajdonságot a konvertáláskor automatikusan hozza létre. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az üres ("") karakterlánc **Null** értékké konvertálását meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

True Az üres ("") karakterlánc **Null** értékké konvertálódik.
False Az üres ("") karakterlánc nem konvertálódik **Null** értékké.

A tulajdonság csak a Visual Basicben állítható be. Ha a beállítást módosítjuk, a módosítás érvényesítéséhez be kell zárnunk, majd ismét meg kell nyitnunk az adatbázist.

A gyorsabb működés érdekében a bármelyik Text vagy Memo mezőbe üres karakterláncot beállító kódot **Null** értéket beállítóra kell módosítani, és a **Properties** halmazból el kell távolítani a **V1xNullBehavior** tulajdonságot.

ValidateOnSet

Szintaxis: *objektum*.**ValidateOnSet** [= *érték*]

Ez a tulajdonság **Field** objektumon érvényes, és a **Field** objektum **Value** tulajdonságának beállításakor az objektum értékének azonnali érvényesítését meghatározó érték kiolvasására vagy beállítására használható. Csak a **Recordset** objektumokban lévő **Fields** objektumok támogatják a **ValidateOnSet** írását és olvasását. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az érték érvényesítését meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

True A tulajdonság a beállításakor érvényesítetté válik.
False A tulajdonság csak a rekord frissítésekor válik érvényessé.

Amikor a **ValidateOnSet** tulajdonság értéke **True**, a **Field** objektum **Value** tulajdonságának módosításakor az objektum **ValidationRule** tulajdonságában meghatározott érvényesítési szabály ellenőrzésre kerül.

Egy **Recordset** objektum tartalmazhat olyan **Fields** objektumgyűjteményt, amelynek számos olyan **Field** objektuma van, amelyek **ValidationRule** tulajdonságának *érték* argumentuma mezők kombinációját érinti. Amikor a **ValidateOnSet** tulajdonság értéke **False**, minden egyes mező **Value** tulajdonságát egymástól függetlenül lehet beállítani, az érvényesítést pedig az utolsó érték beállítás után, a rekord frissítésével lehet végrehajtani.

A **ValidateOnSet** tulajdonság nem támogatott az elavult **Dynaset** és **Table** objektumokkal kapcsolatos **Field** objektumon.

ValidationRule

Szintaxis: *objektum*.**ValidationRule** [= *érték*]

Ez a tulajdonság **Field**, **Recordset** és **TableDef** objektumon érvényes, és a mezőbeli

adat változásakor vagy hozzáadásakor az adatot érvényesítő érték kiolvasására vagy beállítására használható. A tulajdonság **Fields** objektumgyűjteményhez még nem tartozó új objektum esetén írható és olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy **String** típusú karakteres kifejezés, amely az összehasonlítást egy SQL **WHERE** záradék alakjában, de a **WHERE** kulcsszó nélkül határozza meg.

A **ValidationRule** tulajdonság határozza meg, hogy az adatok a kapcsolódó mezőben érvényesek-e. Ha az adatok nem megengedettek, kezelhető hiba következik be. A visszaadott hibaüzenet a **ValidationText** tulajdonság szövege (ha az megadott), vagy a **ValidationRule** által megadott kifejezés lesz.

Field objektumnál a **ValidationRule** tulajdonság használata attól a **Fields** halmazt tartalmazó objektumtól függ, amelyhez a **Field** tartozik:

Ha az objektum:	A tulajdonság használata:
Index	Nem támogatott.
QueryDef	Csak olvasható.
Recordset	Csak olvasható.
Relation	Nem támogatott.
TableDef	Írható és olvasható.

Egy **Recordset** objektum esetén a **ValidationRule** tulajdonság csak olvasható, egy **TableDef** objektumnál pedig az objektum állapotától függően a következő:

Ha a **TableDef**:	A tulajdonság használata:
Alaptábla	Írható és olvasható
Csatolt tábla	Csak olvasható.

Az érvényesítés csak a Microsoft Jet adatbázis-szerkesztőt használó adatbázisok támogatják.

A **Field** objektum **ValidationRule** tulajdonságban adott karakteres kifejezés csak ugyanazon **Field** objektumra hivatkozhat, de nem hivatkozhat felhasználó által definiált függvényekre, tartománybeli összegző függvényekre, SQL összegző függvényekre vagy lekérdezésekre. Egy **Field** objektum **ValidateOnSet** tulajdonságának **True** értéke esetén a **ValidationRule** tulajdonság beállításához a kifejezésnek sikeresen elemzettnek és **True** értékkel kiértékeltnek kell lennie. Ha a **ValidateOnSet** tulajdonság **False**, a **ValidationRule** tulajdonság beállítása figyelmen kívül marad.

Egy **Recordset** vagy **TableDef** objektum **ValidationRule** tulajdonsága az objektumbeli többszörös mezőkre hivatkozhat. A korlátozások a **Field** objektumra vonatkozókkal azonosak.

A tábla típusú **Recordset** objektumokon a **ValidationRule** az objektum létrehozásá-

hoz használt **TableDef** objektum **ValidationRule** tulajdonságának értékét kapja.

Csatolt táblán alapuló **Recordset** objektumokon a **ValidationRule** az alapjául szolgáló tábla értékét kapja. Ha a tábla nem támogatja az érvényesítést, a **ValidationRule** tulajdonság értéke üres ("") karakterlánc lesz.

A **ValidationRule** tulajdonság nem támogatott az elavult **Dynaset** és **Table** objektumokkal kapcsolatos **Field** objektumon.

ValidationText

Szintaxis: *objektum*.**ValidationText** [= *érték*]

Ez a tulajdonság **Field**, **Recordset** és **TableDef** objektumon érvényes, és egy **Field** objektumban a **ValidationRule** szerinti érvényességi feltételeknek nem megfelelő érték esetén az alkalmazásunk által megjelenítendő szöveg kiolvasására vagy beállítására használható. A tulajdonság objektumgyűjteményhez még nem tartozó új objektum, valamint alaptábla esetén írható és olvasható, míg **Recordset** objektum és **TableDef** objektum csatolt táblája esetén csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy érvénytelen adat beírásakor megjelenítendő szöveget tartalmazó **String** típusú karakteres kifejezés.

Field objektumnál a **ValidationText** tulajdonság használata attól a **Fields** halmazt tartalmazó objektumtól függ, amelyhez a **Field** tartozik:

Ha az objektum:	A tulajdonság használata:
Index	Nem támogatott.
QueryDef	Csak olvasható.
Recordset	Csak olvasható.
Relation	Nem támogatott.
TableDef	Írható és olvasható.

Value

Szintaxis: *objektum*.**Value** [= *érték*]

Ez a tulajdonság a **Field**, **Parameter** és **Property** objektumon, illetve a **Column** objektumon és a **CheckBox**, **CommandButton**, **HScrollBar**, **OptionButton** és **VScrollBar** vezérlőelemen eltérő értelmezéssel érvényes.

A **Value** tulajdonság a **Field**, **Parameter** és **Property** objektumon az objektum értékének kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő

objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy objektum Type tulajdonságával megadott adattípusnak megfelelő **Variant** típusú kifejezés.

Általában a **Value** tulajdonság **Recordset** objektumban adatok kiolvasásához és módosításához használatos. A **Value** a **Field**, **Parameter** és **Property** objektum alapértelmezett tulajdonsága.

A **Value** tulajdonság a **Column** objektumon és a **CheckBox**, **CommandButton**, **HScrollBar**, **OptionButton** és **VScrollBar** vezérlőelemen a következő értelmezéssel érvényes:

- **Field** objektumon programfejlesztési fázisban nem elérhető és egy mező tartalmának kiolvasására vagy beállítására használható.
- **CheckBox** és **OptionButton** vezérlőelemen a vezérlőelem állapotának kiolvasására vagy beállítására használható.
- **CommandButton** vezérlőelemen programfejlesztési fázisban nem elérhető és a vezérlőelem kiválasztottságát meghatározó érték kiolvasására vagy beállítására használható.
- **HScrollBar** és **VScrollBar** vezérlőelemen a gördítősáv aktuális helyzetének kiolvasására vagy beállítására használható. A visszaadott érték mindig a gördítősáv **Max** és **Min** tulajdonságainak értékei között van.

A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum egy vezérlőelem állapotát, értékét vagy pozícióját meghatározó érték.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

- **Field** objektumon — Csak a **Field** adattípusai által korlátozott érték.
- **CheckBox** vezérlőelemen:

  0 *Unchecked* — Nem kiválasztott állapotú jelölőnégyzet. (Alapértelmezett.)
  1 *Checked* — Kiválasztott állapotú jelölőnégyzet.
  2 *Grayed* — Nem elérhető jelölőnégyzet.

- **CommandButton** vezérlőelemen:

  True A parancsgomb kiválasztott állapotú. A programon belül szimulált gombnyomás a gombon **Click** eseményt okoz.

False A parancsgomb nem kiválasztott állapotú. (Alapértelmezett.)

- **HScrollBar** és **VScrollBar** vezérlőelemen — A pozíciót meghatározó -32768 és 32767 közötti értékű **Integer** típusú szám.
- **OptionButton** vezérlőelemen:

True A parancsgomb kijelölt állapotú.
False A parancsgomb nem kijelölt állapotú. (Alapértelmezett.)

Egy objektum alapértelmezett tulajdonsága beállított, ezért azt nem kell a programkódban megadni.

Verb

Szintaxis:

Ez a tulajdonság **OLE** befogadó vezérlőelemen érvényes, és egy objektumnak az **Action** tulajdonság használatával történt aktiválásakor végrehajtandó műveletet meghatározó érték kiolvasására vagy beállítására használható, azonban a Visual Basic 4.0-s verziójában csak kompatibilitási okból maradt meg. Helyette a **DoVerb** metódust kell használni.

Version

Szintaxis: *objektum*.**Version**

Ez a tulajdonság **Database** és **DBEngine** objektumon érvényes, és az objektummal kapcsolatos adatbázis vagy adatbázis-szerkesztő verzióját jelző értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **Version** tulajdonság által visszaadott érték az objektumtól függően a következő értékként értelmezett karakterlánc:

- **Database** objektumon — Egy nyitott adatbázis adatformájának verziószáma. Ez az adatbázist létrehozó Jet adatbázis-szerkesztő verziószámát is azonosítja.
- **DBEngine** objektumon — A Microsoft Jet adatbázis-szerkesztő verziószáma.

A **Version** tulajdonság beállításakor a verziószámot a "*nagyobb. kisebb*", például "2.5" alakban kell megadni.

Visible

Szintaxis: *objektum*.**Visible** [= *érték*]

Ez a tulajdonság **Column**, **Form**, **MDIForm** és **Menu** objektumon, valamint

CheckBox, **ComboBox**, **CommandButton**, **CommonDialog**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **Line**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox** és **VScrollBar** vezérlőelemen érvényes, és az objektum vagy vezérlőelem láthatóságát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az objektum láthatóságát vagy rejtettségét meghatározó **Boolean** típusú kifejezés.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

True	Az objektum látható. (Alapértelmezett.)
False	Az objektum rejtett.

Egy objektumot már az indításkor rejtetté tehetünk a **Visible** tulajdonságát a programfejlesztési fázisban **False** értékre beállítva. A tulajdonságot futásidőben használva egy vezérlőelemet rejtetté tehetünk, majd valamely esemény hatására ismét megjeleníthetjük.

A **Show** és **Hide** metódusok űrlapon történő használata azonos az űrlap **Visible** tulajdonságának **True** vagy **False** értékre állításával.

VisibleCols

Szintaxis: *objektum*.**VisibleCols** [= *érték*]

Ez a tulajdonság **DBGrid** vezérlőelemen érvényes, és a vezérlőelem látható oszlopainak számát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum 0 és a **DBGrid** vezérlőelem **Cols** tulajdonságának beállítása közötti **Integer** érték.

A **VisibleCols** tulajdonság a teljesen vagy részben látható **DBGrid** oszlopokat is tartalmazza.

VisibleCount

Szintaxis: *objektum*.**VisibleCount**

Ez a tulajdonság **DBCombo** és **DBList** vezérlőelemen érvényes, és a vezérlőelem látható elemeinek számát adja vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **VisibleCount** tulajdonság 0 és a vezérlőelemben látható elemek száma közötti

értéket ad vissza. Egy elem még akkor is láthatónak minősül, ha szövegének csak egy része látható az **IntegralHeight** tulajdonság **False** beállítása esetén.

VisibleItem

Szintaxis: *objektum*.**VisibleItem**

Ez a tulajdonság **DBCombo** és **DBList** vezérlőelemen érvényes, és a vezérlőelem listájának minden egyes látható eleméhez egy könyvjelzőtömböt ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

Ezek a könyvjelzők a **Recordset** objektumban a lista feltöltésére használt egyedi rekordok elérését teszi lehetővé.

VisibleRows

Szintaxis: *objektum*.**VisibleRows**

Ez a tulajdonság **DBGrid** vezérlőelemen érvényes, és a vezérlőelem látható sorainak számát meghatározó értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **VisibleRows** tulajdonság 0 és a **DBGrid** vezérlőelem **Rows** tulajdonságának beállítása közötti **Integer** értéket ad vissza. A tulajdonság a teljesen vagy részben látható **DBGrid** sorokat is tartalmazza.

Weight

Szintaxis: *objektum*.**Weight** [= *érték*]

Ez a tulajdonság **Font** objektumon érvényes, és az objektumot tördelését meghatározó karaktersúlyok kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a karaktersúlyt meghatározó **Integer** érték.

Mivel a **Font** objektum a programfejlesztési fázisban közvetlenül nem érhető el, **Weight** tulajdonságának értékét a kívánt vezérlőelem Properties ablakban a Font tulajdonság kiválasztásával és a Properties gombra kattintással, majd a Font párbeszédpanel Font Style mezőjében közvetve, a betűtípusok kiválasztásával tehetjük meg. A **Weight** értéke Regular és Italic beállításban 400 (alapértelmezett), a Bold és Bold Italic beállításban pedig 700. A tulajdonság futásidőben történő beállítása közvetlenül, a **Font** objektum számára a szükséges értéket megadva történhet.

Ha futásidőben a **Weight** tulajdonságnak nem 400 vagy 700 értéket adunk, a Visual

Basic azt a hozzá közelebb lévő érvényes értékre konvertálja: 400 és 551 között 400-ra, az 550 felett pedig 700-ra.

WhatsThisButton

Szintaxis: *objektum*.**WhatsThisButton**

Ez a futásidőben csak olvasható tulajdonság **Form** objektumon érvényes, és a What's This gombnak az objektum címsorában történő megjelenését meghatározó értéket ad vissza. A tulajdonság a Windows 95 (vagy Windows NT 3.51) operációs rendszer használatát igényli. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **WhatsThisButton** lehetséges értékei a következők lehetnek:

True A What's This gomb megjelenik a címsorban.
False A What's This gomb nem jelenik meg a címsorban. (Alapértelmezett.)

A **WhatsThisButton** tulajdonság **True** beállításához egyes tulajdonságok beállításainak a következőknek kell lenniük:

Tulajdonság:	Értéke:
WhatsThisHelp	**True**
ControlBox	**True**
BorderStyle	*Fixed Single* vagy *Sizable*
MinButton	**False**
MaxButton	**False**

vagy

Tulajdonság:	Értéke:
BorderStyle	*Fixed Dialog*

WhatsThisHelp

Szintaxis: *objektum*.**WhatsThisHelp**

Ez a futásidőben csak olvasható tulajdonság **Form** és **MDIForm** objektumon érvényes, és a Windows 95 Help vagy a fő Help ablakokban rendelkezésre álló What's This előugró panelnek a környezetérzékeny súgónak általi használatát meghatározó értéket ad vissza. A tulajdonság a Windows 95 (vagy Windows NT 3.51) operációs rendszer használatát igényli. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **WhatsThisHelp** lehetséges értékei a következők lehetnek:

True Az alkalmazás a What's This hozzáférési módszerek valamelyikét használja a Windows Help elindítására és a **WhatsThisHelpID** tulajdonságban adott téma betöltéséhez.

False Az alkalmazás az `F1` billentyűt használja a Windows Help elindítására és a **WhatsThisHelpID** tulajdonságban adott téma betöltéséhez. (Alapértelmezett.)

A **WhatsThisHelp** tulajdonság **True** beállítása esetén a következő What's This súgó hozzáférési módszerek állnak rendelkezésre egy alkalmazás számára:

- Az űrlap címsorában a **WhatsThisButton** használatával egy What's This gomb megjelenítésével. Az egérmutató What's This állapotba (kérdőjeles nyíl) kerül, a megjelenő témát annak a vezérlőelemnek a **WhatsThisHelpID** tulajdonsága határozza meg, amelyikre a felhasználó rákattintott.
- Egy űrlap **WhatsThisMode** metódusának meghívásával. A hatás azonos a What's This gombra történő kattintásával. Ez a metódus például az alkalmazásunk menüsorában egy menü parancsával hívható meg. Egy adott vezérlőelem **ShowWhatsThis** metódusának meghívásával. A megjelenő témát a vezérlőelem **WhatsThisHelpID** tulajdonsága határozza meg.

WhatsThisHelpID

Szintaxis: *objektum*.**WhatsThisHelpID** [= *érték*]

Ez a tulajdonság **CheckBox**, **ComboBox**, **CommandButton**, **Data**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE** befogadó, **OptionButton**, **PictureBox**, **Shape**, **TextBox** és **VScrollBar** vezérlőelemen érvényes, és egy objektumhoz kapcsolódó környezetazonosító kiolvasására vagy beállítására használható. A tulajdonság a Windows 95 (vagy Windows NT 3.51) operációs rendszer használatát igényli. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a súgó környezetazonosítóját meghatározó numerikus típusú kifejezés.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

Ha az *érték*:	Értelmezése:
0	Nincs megadott környezetazonosító. (Alapértelmezett.)
>0	Az objektumhoz kapcsolódó What's This téma érvényes környezetazonosítóját megadó egész szám.

A tulajdonság használatával az alkalmazásunk számára a Windows 95 What's This előugró paneljével környezetérzékeny súgó szolgáltató. A Visual Basic alkalmazások két

különböző modell (Windows 3.x és Windows 95) használatát teszik lehetővé a környezetérzékeny súgó számára.

A Windows 3.x modell az `F1` billentyűt használja a Windows Help elindítására és a **HelpContextID** tulajdonságban adott téma betöltéséhez. A Windows 95 modell jellemzően a jobb felső sarokban megjelenő What's This gombot használja használja a Windows Help elindítására és a **WhatsThisHelpID** tulajdonságban adott téma betöltéséhez. A kétféle modell között a **WhatsThisHelp** tulajdonság beállításával lehet választani.

Width

Szintaxis: *objektum*.**Width** [= *érték*]

A tulajdonság **Columns**, **Form**, **MDIForm**, **Picture**, **Printer** és **Screen** objektumon, továbbá **CheckBox**, **ComboBox**, **CommandButton**, **DBCombo**, **DBGrid**, **DBList**, **DirListBox**, **DriveListBox**, **FileListBox**, **Frame**, **Grid**, **HScrollBar**, **Image**, **Label**, **ListBox**, **OLE befogadó**, **OptionButton**, **PictureBox**, **Shape**, **TextBox** és **VScrollBar** vezérlőelemen az adott objektum vagy vezérlőelem szélességének kiolvasására vagy beállítására használható. A tulajdonság a **Printer** és **Screen** objektumoknál a programfejlesztési fázisban nem érhető el, futásidőben pedig csak olvasható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az adott objektum vagy vezérlőelem szélességét megadó numerikus kifejezés.

A szélesség mérése a következő feltételek szerint történik:

- **Form** objektumon — Az objektum külső (a keretet és címsort is magában foglaló) szélessége twip mértékegységben. 1 (logikai) cm = 567 twip.
- **Picture** objektumon — A kép HiMetric egységben adott szélessége. A HiMetric egység a **ScaleX** és **ScaleY** metódusokkal tetszés szerinti egységre konvertálható.
- **Printer** objektumon — A nyomtatóeszközre beállított valós, fizikai papírméret twip mértékegységben. A programfejlesztési fázisban nem érhető el. Futásidőben a tulajdonság értéke a **PaperSize** tulajdonság beállítása helyett használt.
- **Screen** objektumon — A képernyő szélessége twip mértékegységben. A programfejlesztési fázisban nem érhető el, futásidőben csak olvasható.
- Vezérlőelemeken — A vezérlőelem szélessége a vezérlőelem határának közepétől a különböző szélességekhez megfelelően illeszkedő módon mérve. A tulajdonság a vezérlőelem befogadójának mértékegységét használja.

Az űrlapoknál és vezérlő objektumoknál a **Width** tulajdonság értéke követi azok méretének egy felhasználói művelet vagy programkód okozta változását. A tulajdonság

egyes objektumokra érvényes maximális értéke rendszerfüggő.

A **Width** tulajdonságot a vele összefüggő **Left**, **Top** és **Height** tulajdonságokkal együtt egy objektum vagy vezérlőelem teljes területét érintő művelethez (mozgatásra, méretmódosításra vagy helyszükséglet számítására), a **ScaleLeft**, **ScaleTop**, **ScaleHeight** és **ScaleWidth** tulajdonságokkal együtt pedig a belső területét érintő számításhoz (másik objektum belsejében való rajzolásra vagy mozgatásra) használható.

Ha a **Width** tulajdonságot olyan nyomtatóvezérlőhöz állítjuk be, amely nem engedi meg ezen tulajdonság beállítását, a beállítás hibajelzés nélkül hatástalan lesz. A csak bizonyos értékeket elfogadó nyomtatóvezérlő esetén a beállítás hibajelzés nélkül az előírt érték lesz.

A **DBGrid** vezérlőelem **Columns** objektumánál a **Width** tulajdonság a **DBGrid** befogadó objektuma által meghatározott egységben adott. Az alapértelmezett érték a **DBGrid** vezérlőelem **DefColWidth** tulajdonságának megfelelő lesz.

WindowList

Szintaxis: *objektum*.**WindowList**

Ez a futásidőben csak olvasható tulajdonság **Menu** objektumon érvényes, és egy **MDIForm** objektumban az aktuális MDI-alablakok egy listájának a **Menu** objektum általi karbantartását meghatározó értéket ad vissza. A tulajdonság szintaxisában az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.

A **WindowList** lehetséges értékei a következők lehetnek:

True A **Menu** objektum karbantartja egy nyitott ablak listáját és az aktív ablak mellett egy pipát jelenít meg. A felhasználó az ablakot annak nevére kattintva aktiválhatja.

False A **Menu** objektum nem tartja karban egy nyitott ablak listáját. (Alapértelmezett.)

Számos MDI-alkalmazásnak (például a Microsoft Exel vagy a Microsoft Word for Windows) van a nyitott MDI-alablakok listáját is tartalmazó Window menüje. A **WindowList** tulajdonság lehetővé teszi az ilyen megjelenésű és viselkedésű alkalmazások készíthetők.

Egy űrlapon csak egy **Menu** objektum **WindowList** tulajdonsága lehet **True** értékű. A tulajdonság beállításakor az MDI-alűrlapok listája a meglévő menü alatt jelenik meg.

Amikor a Menu Editor párbeszédpanelen a Window List jelölőnégyzetet kijelölté tesszük egy **Menu** objektumhoz, a létrehozott menühöz a nyitott MDI-alablak listája is megjelenik.

WindowState

Szintaxis: *objektum*.**WindowState** [= *érték*]

A tulajdonság **Form** és **MDIForm** objektumon érvényes, és egy űrlap ablakának futásidejű megjelenési állapotát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum az objektum állapotát meghatározó **Integer** típusú érték.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

Ha az *érték*:	Az űrlap megjelenése:
0	*Normal* — Normál. (Alapértelmezett.)
1	*Minimized* — Ikonállapotú (ikonná zsugorított).
2	*Maximized* — Teljes méretű (a legnagyobb méretben megjelenített).

Mielőtt az űrlap megjelenik, a **WindowState** tulajdonsága 0 (*Normal*), függetlenül annak kezdeti beállításától. Ez az állapot az űrlap **Height**, **Left**, **ScaleHeight**, **ScaleWidth**, **Top** és **Width** tulajdonságainak értékében is érvényesül. Ha az űrlap a megjelenítés után rejtetté válik, ezek a tulajdonságok a következő megjelenítésig megtartják értékeiket, függetlenül az időközben a **WindowState** tulajdonság értékében bekövetkezett változástól.

WordWrap

Szintaxis: *objektum*.**WordWrap** [= *érték*]

A tulajdonság **Label** vezérlőelemen érvényes, és az **AutoSize** = **True** beállítású **Label** vezérlőelem függőleges vagy vízszintes méretnek a **Caption** tulajdonságában megadott szövegmérethez illeszkedését meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a **Label** vezérlőelem szövegmérethez illeszkedését meghatározó **Boolean** típusú érték.

Az *érték* argumentum lehetséges értékei a következők lehetnek:

True A szöveg tördelt lesz, a **Label** vezérlőelem függőlegesen a szövegméretnek megfelelően terjed ki vagy húzódik össze.

False A szöveg nem lesz tördelt, a **Label** vezérlőelem a vízszintesen a szöveghossz által kívánt mértékben terjed ki vagy húzódik össze, a függőleges méretét a karakternagyság és a sorok száma határozza meg.

A tulajdonsággal a **Label** vezérlőelem tartalmának megjelenési formáját írhatjuk elő. Ha a **Label** vízszintes méretének állandónak tartása mellett eltérő szövegméret engedélyezéséhez a **WordWrap** és az **AutoSize** tulajdonságnak is **True** értéket kell adni. A **Label** csak vízszintes méretének változását a **WordWrap** tulajdonság **False** értéke engedélyezi. Ha állandó méretű **Label** vezérlőelemre van szükségünk, az **AutoSize** tulajdonságot **False** értékre kell beállítani. Ebben az esetben a szöveg mindig tördelődik, függetlenül a **Label** méretétől vagy a **WordWrap** tulajdonság értékétől, vagyis a címke méreténél nagyobb helyigényű szövegek megcsonkulnak, hiszen a címke mérete nem változhat.

X1

Szintaxis: *objektum*.**X1** [= *érték*]

A tulajdonság **Line** vezérlőelemen érvényes, és a vezérlőelem kezdőpontjának vízszintes koordinátáját meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a vízszintes koordinátát meghatározó numerikus érték.

Az **X1**, **Y1**, **X2**, **Y2** tulajdonságok együttes használatával futásidőben dinamikusan kiterjeszthető a **Line** vezérlőelem az **X1**,**Y1** ponttól az **X2**,**Y2** pontig, így például két lista elemei vagy egy kép pontjai közötti kapcsolatot szemléltetve.

X2

Szintaxis: *objektum*.**X2** [= *érték*]

A tulajdonság **Line** vezérlőelemen érvényes, és a vezérlőelem végpontjának vízszintes koordinátáját meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság alkalmazására az **X1** tulajdonságnál leírtak értelemszerűen érvényesek.

Y1

Szintaxis: *objektum*.**Y1** [= *érték*]

A tulajdonság **Line** vezérlőelemen érvényes, és a vezérlőelem kezdőpontjának függőleges koordinátáját meghatározó érték kiolvasására vagy beállítására használható. A

tulajdonság alkalmazására az **X1** tulajdonságnál leírtak értelemszerűen érvényesek.

Y2

Szintaxis: *objektum*.**Y1** [= *érték*]

A tulajdonság **Line** vezérlőelemen érvényes, és a vezérlőelem végpontjának függőleges koordinátáját meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság alkalmazására az **X1** tulajdonságnál leírtak értelemszerűen érvényesek.

Zoom

Szintaxis: *objektum*.**Zoom** [= *érték*]

Ez a programfejlesztési fázisban nem elérhető tulajdonság **Printer** objektumon érvényes, és a nyomtatás százalékos nagyítási vagy kicsinyítési arányát meghatározó érték kiolvasására vagy beállítására használható. A tulajdonság szintaxisa a következő objektumazonosítót és argumentumot tartalmazza:

- Az *objektum* azonosítója egy objektumot kijelölő objektumkifejezés.
- A tulajdonság beállításához használható *érték* argumentum a nyomtatás százalékos skálázási arányát meghatározó numerikus kifejezés. Az *érték* alapértelmezetten 0, amely normális méretű oldal nyomtatását jelenti.

A **Zoom** tulajdonság beállítása a fizikailag nyomtatott oldalméretet nagyítja vagy kicsinyíti **Zoom**/100 lépcsőben egészen a látszólagos nyomtatási méretig.

A **Printer** objektum tulajdonságainak hatása a nyomtató gyártója által adott vezérlőprogramtól függ: egyes tulajdonságok hatástalanok lehetnek vagy több tulajdonság azonos hatást eredményezhet. Erre vonatkozóan a nyomtató leírása tartalmazhat további információt.

Fordítási direktívák

#Const

Szintaxis: **#Const** *konstansnév = kifejezés*

Ez a direktíva feltételes fordítási konstanst definiál a Visual Basic számára. A direktíva szintaxisa a következő részeket tartalmazza:

- A *konstansnév* a definiálandó konstansnak a szokásos névadási konvenciónak megfelelő neve
- A *kifejezés* egy betűszerinti, más feltételes fordítási konstans vagy ezek bármely

aritmetikai vagy — az **Is** kivételével — bármely logika operátor kombinációja.

A feltételes fordítási konstans mindig **Private** arra a modulra nézve, amelyben megjelenik. Ezzel a direktívával nem lehetséges **Public** fordítási konstans létrehozása, az csak a felhasználói felületen lehetséges.

A *kifejezés* csak feltételes fordítási konstansokat (és betűszerinti kifejezéseket) tartalmazhat. A standard (a **Const** utasítással létrehozott) vagy definiálatlan konstansok használata hibát okoz. Hasonlóképpen, a **#Const** direktívával létrehozott konstans is csak a feltételes fordításhoz használható.

A feltételes fordítási konstansok — a programkódbeli helyüktől függetlenül — mindig modulszinten értékelődnek ki.

#If...Then...#Else#If...Then...#Else

Szintaxis:

#If *feltétel1* **Then**
    utasításcsoport1
[**#ElseIf** *feltétel-n* **Then**
    [*utasításcsoport2*]]
...
[**#Else**
    [*utasításcsoport n*]]
#End If

Ez az utasítás feltételesen fordítja a kiválasztott utsításcsoportokat, azok közül a feltételektől függően csak az egyik kerül fordításra.

- A *feltétel1* bármely, egy vagy több feltételes fordítási konstanst, betűszerinti kifejezést és operátort tartalmazó, **True** vagy **False** eredményt adó kifejezés.
- Az *utasításcsoport1* Visual Basic programsorokat vagy fordítási direktívákat tartalmazhat, amelyek a *feltétel1* **True** értéke esetén kell végrehajtani.
- Ha a *feltétel1* kiértékelése **False** eredményt adott, a feltételes **#ElseIf** *feltétel-n* **Then** argumentumban szereplő kifejezés értékétől függően kerül végrehajtásra a következő, feltételes *utasításcsoport2*. Ha a teljes láncban az **#If** és **#ElseIf** utáni feltételek egyike sem teljesül, a feltételes **#Else** *utasításcsoport_n* utasítás végrehajtása után az **#End If** utasítást követő sorba kerülünk.

Az **#If...Then...#Else** direktíva működése azonos az **If...Then...Else** utasításéval, azzal az eltéréssel, hogy az **#If**, **#Else**, **#ElseIf** és **#EndIf** direktíváknak nincs egysoros alakja, továbbá velük azonos sorban csak direktíva jelenhet meg.

A feltételes fordítás jellemzően ugyanazon program különböző platformokon történő fordítására, és a futáskövetéshez írt kódrészleteknek a futtatható fájlba történő fordításának megakadályozására használatos. Mivel a feltételes fordítás alatti programrész a

végleges futtatható fájl fordításakor teljesen figyelmen kívül marad, nincs hatása annak méretére vagy teljesítményére sem.

A direktívában mindegyik feltétel kiértékelésre kerül, függetlenül az előzők eredményétől, ezért a feltételekben használt minden konstansnak definiáltnak kell lennie, a definiálatlan konstans **Empty** értékként értékelődik.

Az **Option Compare** utasítás hatástalan az **#If** és **#ElseIf** utasításban használt kifejezésekre. A feltételes fordítási direktívákban szereplő kifejezés mindig az **Option Compare Text** összehasonlítási móddal értékelődik.

SQL nyelvi elemek összefoglalása

Függvények

Avg()

Szintaxis: **Avg(** *kifejezés* **)**

Ez az SQL függény egy lekérdezésnek a *kifejezés* argumentumban szereplő karakteres kifejezéssel megadott mezőjében lévő numerikus értékhalmaz számtani átlagát határozza meg. Ha a *kifejezés* állandót vagy függvényt tartalmaz, az **Avg( )** függvény a mező értékein végrehajtja a kifejezésnek megfelelő műveletet. A *kifejezés* táblamezőt és — más SQL függvény kivételével — bármely Visual Basic beépített függvényt vagy felhasználó által definiált függvényt is tartalmazhat. Az **Avg( )** függvény az átlagszámítás során figyelmen kívül hagyja a **Null** mezőket.

A függvény nemcsak lekérdező kifejezésekben, de egy **QureyDef** objektum **SQL** tulajdonságában vagy egy SQL lekérdezésen alapuló **Recordset** objektum létrehozásában is használható.

Count( )

Szintaxis: **Count(** *kifejezés* | ***)**

Ez az SQL függény egy lekérdezésnek a *kifejezés* argumentumban szereplő karakteres kifejezéssel megadott mezőjében lévő rekordok számát — a rekordok tárolt értékektől függetlenül — határozza meg. Ha a *kifejezés* állandót vagy függvényt tartalmaz, a **Count( )** függvény a mező értékein végrehajtja a kifejezésnek megfelelő műveletet. A *kifejezés* táblamezőt és — más SQL függvény kivételével — bármely Visual Basic beépített függvényt vagy felhasználó által definiált függvényt is tartalmazhat.

A függvény minden adattípus, akár szöveges rekordok számolására is használható. A függvény a számolás során a **Null** mezőket tartalmazó rekordokat figyelmen kívül hagyja, kivéve, ha a **Count(*)** alakot használjuk: ebben az esetben a teljes rekordszámot

kapjuk eredményül, ráadásul a függvény lényegesen gyorsabbá is válik. Ha a *kifejezés* több, "&" karakterrel elválasztott mezőt jelöl ki, a **Count( )** csak azokat a mezőket számolja, amelyeknek legalább egy mezője nem **Null**. Ha minden mező **Null**, a rekord nem kerül számolásra.

A függvény nemcsak lekérdező kifejezésekben, de egy **QureyDef** objektum **SQL** tulajdonságában vagy egy SQL lekérdezésen alapuló **Recordset** objektum létrehozásában is használható parancsán keresztül lehet módosítani.

First( )

Szintaxis: **First(** *kifejezés* **)**

Ez az SQL függény egy lekérdezés által visszaadott eredményhalmaz első rekordjából a *kifejezés* argumentumban szereplő karakteres kifejezéssel megadott mező egy mezőértéket adja meg. Ha a *kifejezés* állandót vagy függvényt tartalmaz, a **First( )** függvény a mező értékein végrehajtja a kifejezésnek megfelelő műveletet. A *kifejezés* táblamezőt és — más SQL függvény kivételével — bármely Visual Basic beépített függvényt vagy felhasználó által definiált függvényt is tartalmazhat.

A **First( )** függvény a **Recordset** objektum **MoveFirst** metódusával analóg funkció. A függvény használatakor ne felejtsük el, hogy a rekordok visszaadási sorrendje csak a lekérdezésben beállított **ORDER BY** záradék esetén lesz meghatározott, egyébként teljesen esetleges, hogy függvény melyik rekordot fogja használja.

Last( )

Szintaxis: **Last(** *kifejezés* **)**

Ez az SQL függény egy lekérdezés által visszaadott eredményhalmaz utolsó rekordjából a *kifejezés* argumentumban szereplő karakteres kifejezéssel megadott mező egy mezőértéket adja meg. Ha a *kifejezés* állandót vagy függvényt tartalmaz, a **Last( )** függvény a mező értékein végrehajtja a kifejezésnek megfelelő műveletet. A *kifejezés* táblamezőt és — más SQL függvény kivételével — bármely Visual Basic beépített függvényt vagy felhasználó által definiált függvényt is tartalmazhat.

A **Last( )** függvény a **Recordset** objektum **MoveLast** metódusával analóg funkció. A függvény használatakor ne felejtsük el, hogy a rekordok visszaadási sorrendje csak a lekérdezésben beállított **ORDER BY** záradék esetén lesz meghatározott, egyébként teljesen esetleges, hogy függvény melyik rekordot fogja használja.

Max( )

Szintaxis: **Max(** *kifejezés* **)**

Ez az SQL függény a *kifejezés* argumentumban szereplő karakteres kifejezéssel

megadott mezőben lévő értékhalmaz maximumát határozza meg. Ha a *kifejezés* állandót vagy függvényt tartalmaz, a **Max( )** a mező értékein végrehajtja az adott kifejezésnek megfelelő műveletet. A *kifejezés* táblamezőt és — más SQL függvény kivételével — bármely Visual Basic beépített függvényt vagy felhasználó által definiált függvényt is tartalmazhat.

A függvényt meghatározott felhalmozáson vagy csoportosításon alapuló mező legnagyobb értékének meghatározására használhatjuk. Felhalmozás hiányában a függvény a teljes táblát használja.

A függvény nemcsak lekérdező kifejezésekben, de egy **QureyDef** objektum **SQL** tulajdonságában vagy egy SQL lekérdezésen alapuló **Recordset** objektum létrehozásában is használható.

Min()

Szintaxis: **Min(** *kifejezés* **)**

Ez az SQL függény a *kifejezés* argumentumban szereplő karakteres kifejezéssel megadott mezőben lévő numerikus értékhalmaz minimumát határozza meg. Ha a *kifejezés* a tábla mezőneve mellett állandót vagy függvényt tartalmaz, a **Min( )** a mező értékein végrehajtja az adott kifejezésnek megfelelő műveletet. A *kifejezés* táblamezőt és — más SQL függvény kivételével — bármely Visual Basic beépített függvényt vagy felhasználó által definiált függvényt is tartalmazhat.

A függvényt meghatározott felhalmozáson vagy csoportosításon alapuló mező legkisebb értékének meghatározására használhatjuk. Felhalmozás hiányában a függvény a teljes táblát használja.

A függvény nemcsak lekérdező kifejezésekben, de egy **QureyDef** objektum **SQL** tulajdonságában vagy egy SQL lekérdezésen alapuló **Recordset** objektum létrehozásában is használható.

StDev()

Szintaxis: **StDev**(*kifejezés*)

Ez az SQL függény a *kifejezés* argumentumban szereplő karakteres kifejezéssel megadott mezőben lévő numerikus értékhalmazzal adott populációból vett minta standard szórását határozza meg. Ha a *kifejezés* állandót vagy függvényt tartalmaz, az **StDev( )** függvény a mező értékein végrehajtja az adott kifejezésnek megfelelő műveletet. A *kifejezés* táblamezőt és — más SQL függvény kivételével — bármely Visual Basic beépített függvényt vagy felhasználó által definiált függvényt is tartalmazhat. Ha a lekérdezés csak egy rekordot tartalmaz, a függvény eredménye **Null** lesz.

A függvény nemcsak lekérdező kifejezésekben, de egy **QureyDef** objektum **SQL** tulajdonságában vagy egy SQL lekérdezésen alapuló **Recordset** objektum létrehozásában

is használható.

StDevP()

Szintaxis: **StDevP**(*kifejezés*)

Ez az SQL függény a *kifejezés* argumentumban szereplő karakteres kifejezéssel megadott mezőben lévő értékhalmazzal adott populáció standard szórását határozza meg. Ha a *kifejezés* állandót vagy függvényt tartalmaz, az **StDevP( )** függvény a mező értékein végrehajtja az adott kifejezésnek megfelelő műveletet. A *kifejezés* táblamezőt és — más SQL függvény kivételével — bármely Visual Basic beépített függvényt vagy felhasználó által definiált függvényt is tartalmazhat.

Ha a lekérdezés nem tartalmaz rekordot, a függvény eredménye **Null** lesz.

A függvény nemcsak lekérdező kifejezésekben, de egy **QureyDef** objektum **SQL** tulajdonságában vagy egy SQL lekérdezésen alapuló **Recordset** objektum létrehozásában is használható.

Sum()

Szintaxis: **Sum**(*kifejezés*)

Ez az SQL függény egy lekérdezésnek a *kifejezés* argumentumban szereplő karakteres kifejezéssel megadott mezőjében lévő numerikus értékhalmaz összegét határozza meg. Ha a *kifejezés* állandót vagy függvényt tartalmaz, a **Sum** függvény a mező értékein végrehajtja az adott kifejezésnek megfelelő műveletet. A *kifejezés* táblamezőt és — más SQL függvény kivételével — bármely Visual Basic beépített függvényt vagy felhasználó által definiált függvényt is tartalmazhat. A **Null** mezőket tartalmazó rekordokat a függvény figyelmen kívül hagyja.

A függvény nemcsak lekérdező kifejezésekben, de egy **QureyDef** objektum **SQL** tulajdonságában vagy egy SQL lekérdezésen alapuló **Recordset** objektum létrehozásában is használható.

Var()

Szintaxis: **Var**(*kifejezés*)

Ez az SQL függény a *kifejezés* argumentumban szereplő karakteres kifejezéssel megadott mezőben lévő numerikus értékhalmazzal adott populációból vett minta varianciabecslését határozza meg. Ha a *kifejezés* állandót vagy függvényt tartalmaz, a **Var( )** függvény a mező értékein végrehajtja az adott kifejezésnek megfelelő műveletet. A *kifejezés* táblamezőt és — más SQL függvény kivételével — bármely Visual Basic beépített függvényt vagy felhasználó által definiált függvényt is tartalmazhat. Ha a lekérdezés csak egy rekordot tartalmaz, a függvény eredménye **Null** lesz.

A függvény nemcsak lekérdező kifejezésekben, de egy **QureyDef** objektum **SQL** tulajdonságában vagy egy SQL lekérdezésen alapuló **Recordset** objektum létrehozásában is használható.

VarP( )

Szintaxis: **VarP**(*kifejezés*)

Ez az SQL függény a *kifejezés* argumentumban szereplő karakteres kifejezéssel megadott mezőben lévő értékhalmazzal adott populáció varianciabecslését határozza meg. Ha a *kifejezés* állandót vagy függvényt tartalmaz, a VarP( ) függvény a mező értékein végrehajtja az adott kifejezésnek megfelelő műveletet. A *kifejezés* táblamezőt és — más SQL függvény kivételével — bármely Visual Basic beépített függvényt vagy felhasználó által definiált függvényt is tartalmazhat.

Ha a lekérdezés nem tartalmaz rekordot, a függvény eredménye **Null** lesz.

A függvény nemcsak lekérdező kifejezésekben, de egy **QureyDef** objektum **SQL** tulajdonságában vagy egy SQL lekérdezésen alapuló **Recordset** objektum létrehozásában is használható.

Utasítások

ALTER TABLE

Szintaxis:

ALTER TABLE *tábla* { **ADD** { **COLUMN** *mezőtípus* [(*méret*)]
[**CONSTRAINT** *index*] |
CONSTRAINT *multiindex* } |
DROP {**COLUMN** *mező* |
CONSTRAINT *indexnév*}}

Ezzel az SQL utasítással a **CREATE TABLE** utasítással létrehozott tábla megjelenését lehet módosítani. Mint látható, az utasítás négy alakban is használható. Az utasítás lehetséges argumentumai a következők:

- A *tábla* argumentumban a módosítandó tábla nevét kell megadni.
- A *mező* argumentumban a *tábla* módosítandó mezőjét jelöljük ki.
- A mező adattípusát a *típus* argumentum határozza meg.
- A feltételes *méret* argumentumban a **Text** vagy **Binary** típusú mezők karakterekben mért hosszát adhatjuk meg.
- A feltételes **CONSTRAINT** záradék *index* argumentuma a *táblá*-hoz adandó vagy onnan törlendő *mező* indexét, illetve a *multiindex* a többszörös mezők in-

dexét határozza meg.

Bármely alakot is használjuk, egy utasítással csak egy mezőt vagy indexet lehet a táblához adni vagy törölni. Egy táblához egyszeres vagy többszörös mezők indexét a **CREATE INDEX** utasítással lehet létrehozni. Az **ALTER TABLE** vagy **CREATE INDEX** utasításokkal létrehozott index törlésére az **ALTER TABLE** vagy **DROP** utasítások egyaránt alkalmazhatók.

A Microsoft Jet adatbázis-szerkesztő nem támogatja az **ALTER TABLE** vagy bármely DDL utasítást a nem Jet adatbázisokkal, helyettük a megfelelő **Create** metódusok használhatók.

CREATE INDEX

Szintaxis:

CREATE [UNIQUE] INDEX *index*
ON *tábla* (*mező* [**ASC** | **DESC**] [, *mező* **ASC** | **DESC**] , ...])
[**WITH** { **PRIMARY** | **DISALLOW NULL** | **IGNORE NULL**}]

Ezzel az SQL utasítással egy létező táblán egy új indexet lehet létrehozni. Az utasítás lehetséges argumentumai a következők:

- Az *index* argumentumban a létrehozandó index nevét kell megadni.
- A *tábla* argumentumban annak a táblának a nevét kell megadni, amelyen az indexet létre kívánjuk hozni.
- A *mező* argumentum azt a mezőnevet (vagy neveket) jelöli ki, amelyet indexelni kell. Egymezős index létrehozásához a táblanév után a mezőnevet zárójelek között kell megadni. A többmezős index létrehozásához minden egyes táblanévnek az indexben kell szerepelnie.

Az utasításban több feltételes kulcsszó található, amelyek a következőképpen befolyásolják az utasítás működését:

- A **UNIQUE** kulcsszó alkalmazásával megelőzhető a rekordok indexelt mezőjén vagy mezőin többszörös értékek létrehozása.
- Az indexelés irányát a feltételes **ASC** kulcsszóval növekvő, a **DESC** kulcsszóval csökkenő irányba lehet beállítani.
- Az ugyancsak feltételes **WITH** kulcsszóval az adatok érvényességi szabályai írhatók elő:
  - A **WITH PRIMARY** alak az indexelt mezőt vagy mezőket elsődleges kulcsnak állítja be. Mivel ez egyben egyedi kulcsot is jelent, használata mellett a **UNIQUE** kucsszó elhagyható. Már létező elsődleges kulcsú táblán új index létrehozásakor a **PRIMARY** használata hibát okoz.
  - A **WITH DISALLOW NULL** alak kizárja az új rekordok indexelt mezőjé-

ből vagy mezőiből a **Null** bejegyzéseket.

- A **WITH IGNORE NULL** alak kizárja az indexelt mező vagy mezők **Null** értéket tartalmazó rekordjait az indexek közül.

A **CREATE INDEX** utasítással egy már indexelt ODBC adatforrás, például SQL Server csatolt tábláin lehet pszeudo-indexelést létrehozni. A pszeudó-index létrehozása a távoli kiszolgálóhoz történő hozzáférés vagy engedély nélkül is lehetséges, a távoli adatbázis és a pszeudó-index nem hathatnak egymásra. Az eredeti és csatolt táblák esetén ugyanazon szintaxis használható. Az utasítás különösen az index hiányában csak olvasható táblán történő indexelés létrehozása esetén előnyös.

Az **ALTER TABLE** utasítással ugyancsak lehetséges egy egyszeres vagy többszörös mezőindexet adni egy táblához, míg az **ALTER TABLE** vagy **CREATE INDEX** utasításokkal létrehozott index törlésére az **ALTER TABLE** vagy **DROP** utasítások egyaránt alkalmazhatók.

A Microsoft Jet adatbázis-szerkesztő nem támogatja a **CREATE INDEX** vagy bármely DDL utasítást a nem Jet adatbázisokkal, helyettük a megfelelő **Create** metódusok használhatók.

CREATE TABLE

Szintaxis:

CREATE TABLE *tábla* (*mező1 típus* [(*méret*)] [*index1*]
[, *mező2 típus* [(*méret*)] [*index2*] [, ...]] [, *multiindex* [, ...]])

Ezzel az SQL utasítással egy új tábla hozható létre. Az utasítás lehetséges argumentumai a következők:

- A *tábla* argumentumban a létrehozandó tábla nevét kell megadni.
- A *mező* argumentumok az új táblán érvényes mezőneveket jelölik ki. Legalább egy mezőnévnek megadottnak kell lennie.
- A *típus* argumentumban az új tábla mezőinek adattípusa szerepel.
- A feltételes *méret* argumentumokban a **Text** és **Binary** mezők karakterben mért nagyságát írhatjuk elő.
- A feltételes *index* argumentumokban a **CONSTRAINT** záradékkal definiált egymezős indexek adható meg.
- Az ugyancsak feltételes *multiindex* argumentumban a **CONSTRAINT** záradékkal definiált többmezős index adható meg.

A **CREATE TABLE** utasítás általában az Execute metódussal használatos.

A Microsoft Jet adatbázis-szerkesztő nem támogatja a **CREATE TABLE** vagy bármely DDL utasítást a nem Jet adatbázisokkal, helyettük a megfelelő **Create** metódusok használhatók.

DELETE

Szintaxis: **DELETE** [*tábla.**] **FROM** *táblakifejezés* **WITH** *feltétel*

Ez az SQL utasítás egy törlő lekérdezést hoz létre, amely egy vagy több tábla adott feltételnek megfelelő rekordjait törli. Az utasítás lehetséges argumentumai a következők:

- A feltételes — de többszörös táblát tartalmazó **DELETE** utasításban kötelező — *tábla* argumentumban annak a táblának a nevét kell megadni, amelyről rekordokat akarunk törölni.
- A **FROM** *táblakifejezés* annak a táblának vagy tábláknak a nevét adja meg, amelyről rekordokat akarunk törölni. Az argumentum egyetlen táblanév vagy az **INNER JOIN**, **LEFT JOIN** vagy **RIGHT JOIN** műveletek eredményeként kapott névkeverék lehet.
- A törlendő rekordokat a **WITH** *feltétel* határozza meg.

A **DELETE** utasítás elsősorban olyan esetekben hasznos, ha sok rekordot kell törölni.

Ha egy tábla mindegyik rekordját töröljük, az üres tábla struktúrája és tulajdonságai továbbra is változatlanul megmaradnak. Ha nem akarjuk a táblastruktúrát megőrizni, hatékonyabb lehet magának a teljes táblának a törlése, például az **Execute** metódust a **DROP TABLE** utasítással használva.

A **DELETE** utasítással egyetlen tábla vagy egy-többre kapcsolat több oldaláról is törölhetünk rekordokat. Egy lekérdezésben a kaszkád törlési műveletek az egy-többre kapcsolat többre oldaláról törölnek.

Ez a törlő lekérdezés csak teljes rekordot töröl, adott mező törlése **Null** értékűre változtató frissítő lekérdezéssel lehetséges.

Mivel a törlő lekérdezés hatása nem vonható vissza, célszerű a végleges törlés előtt egy ugyanezen *feltétel* szerinti választó lekérdezés eredménye alapján ellenőrizni a törlésre kerülő rekordokat, továbbá legalább a törlés előtt adatainkról biztonsági másolatot készíteni.

DROP

Szintaxis: **DROP** {**TABLE** *tábla* | **INDEX** *index* **ON** *tábla*}

Ez az SQL utasítás egy adatbázisból töröl egy létező táblát vagy egy tábla létező indexét. Az utasítás két lehetséges alakjának argumentumai a következők:

- A törölni kívánt tábla nevét a **TABLE** *tábla* argumentumban, a törölni kívánt indexet tartalmazó tábla nevét pedig az **ON** *tábla* argumentumban kell megadni.
- A tábláról törölni kívánt indexet az **INDEX** *index* argumentumban kell megadni.

A tábla vagy a tábla egy indexének törlése előtt a táblát le kell zárni. Tábláról inde-

xet az **ALTER TABLE** utasítással is törölni lehet. Általában a **DROP** utasítás az **Execute** metódussal használatos.

A Microsoft Jet adatbázis-szerkesztő nem támogatja a **DROP** vagy bármely DDL utasítást a nem Jet adatbázisokkal, helyettük a megfelelő **Create** metódusok használhatók.

INSERT INTOINSERT INTO

Szintaxis 1:

INSERT INTO *cél* [(*mező1* [, *mező2* [, ...]])]
VALUES (*érték1* [, *érték2* [, ...])

Szintaxis 2:

INSERT INTO *cél* [**IN** *külső_adatbázis*] [(*mező1* [, *mező2* [, ...]])]
SELECT [*forrás.*]*mező1* [, *mező2* [, ...]
FROM *táblakifejezés*

Ez a hozzáfűző lekérdezést megvalósító SQL utasítás egy vagy több rekordot ad egy tábla végéhez.

- Egyetlen rekord hozzáadása esetén az **INSERT INTO** utasítás a Szintaxis 1 szerinti felépítésű,és az egyes argumentumok értelmezése a következő:
  - Annak a táblának vagy lekérdezésnek a nevét, amelyhez a rekordot hozzá kívánjuk adni, a *cél* argumentumban kell megadni.
  - A feltételes *mező* argumentumok azon mezők nevének listáját adják, amelyekhez adatokat kívánunk adni.
  - A feltételes *érték* argumentumok azon értékek listáját adják, amelyeket az új rekord adott mezőiben kívánunk elhelyezni. Az egyes értékek az értéklistán belüli pozíciójuknak megfelelő mezőbe kerülnek.
- Többszörös rekordok, rekordhalmazok hozzáadáshoz az **INSERT INTO** utasítást a Szintaxis 2 szerint kell használni. Ekkor az argumentumok értelmezése a következő:
  - Annak a táblának vagy lekérdezésnek a nevét, amelyhez a rekordot hozzá kívánjuk adni, a *cél* argumentumban kell megadni.
  - Annak a táblának vagy lekérdezésnek a neve, amelyből a rekordot másolni kívánjuk, a feltételes *forrás* argumentumban szerepel.
  - Ha egy külső adatbázishoz akarunk rekordokat adni, annak elérési útját a feltételes *külső_adatbázis* argumentumban kell megadni.
  - A feltételes *mező* argumentumok a *cél* után azon mezők nevének listáját adják meg, amelyekhez adatokat kívánunk adni, a *forrás* után pedig azon

mezők nevének listája, amelyekből adatokat kívánunk nyerni.

- A *táblakifejezés* argumentumokban azoknak a tábláknak a neve szerepel, amelyből a rekordok beillesztése történik. Ez egyetlen táblanév, egy **INNER JOIN**, **LEFT JOIN** vagy **RIGHT JOIN** műveletet, vagy tárolt lekérdezést eredményező névkeverék lehet.

Egyetlen új rekord hozzáadásakor (Szintaxis 1) az **INSERT INTO** utasításban a rekordmezőket és azok értékeit hiánytalanul kell megadni. Ha nincs minden egyes mező specifikálva, a hiányzó oszlopokba az alapértelmezett vagy a **Null** érték kerül. Ha a feltételes mezőlista hiányzik, az értéklista a tábla minden mezőjéhez kell értéket tartalmaznia, különben az utasítás végrehajtása sikertelen lesz. Ezt a szintaxist akkor használhatjuk, ha meglévő rekord másolata helyett egyedi értékekel definiált rekordot akarunk egy táblához adni

A többszörös rekordok, rekordhalmazok hozzáadása (Szintaxis 2), esetén már létező rekord vagy rekordhalmaz hozzáadása lehetséges: a [**IN** *külső_adatbázis*] [(*mező1* [, *mező2* [, ...]])] részben kijelölt — esetleg külső adatbázisban található — táblamezőkbe az ugyancsak feltételes **FROM** *táblakifejezés* rész által adott táblából vagy táblákból a **SELECT** [*forrás.*]*mező1* [, *mező2* [, ...] résszel kijelölt mezők másolatát adjuk hozzá. A kiválasztott rekordok csak az egyik tábláról a másikra másolódnak, az utasításnak az eredeti rekordot tartalmazó táblára nincs hatása.

A *cél* vagy *forrás* egy táblát vagy lekérdezést határozhat meg. A lekérdezés esetén a Mirosoft Jet a lekérdezés által specifikált összes táblához elvégzi a hozzáadást.

Ha a *cél* tábla elsődleges kulcsot tartalmaz, gondoskodni kell arról, hogy annak mezőjéhez vagy mezőihez egyedi, nem **Null** értékek adódjanak, ellenkező esetben a Microsoft Jet nem végzi el a rekordok hozzáadását.

A rekordokat egy **Counter** mezővel rendelkező táblához adva, a lekérdezés nem tartalmazhatja a **Counter** mezőt.

A hozzáfűző lekérdezés végrehajtása előtt a hozzáadandó rekordok egy azonos kiválasztási feltételű **SELECT** választó lekérdezés eredménye alapján ellenőrízhetők.

SELECT

Szintaxis:

SELECT [*állítás*] { * | *tábla.* * | [*tábla.*]*mező1* [, [*tábla.*]*mező2* [, ...]] }
[**AS** *hivatkozási_név1* [, *hivatkozási_név2* [, ...]]]
FROM *táblakifejezés* [, ...] [**IN** *külső_adatbázis*]
[**WHERE** ...]
[**GROUP BY**...]
[**HAVING**...]
[**ORDER BY**...]
[**WITH OWNERACCESS OPTION**]

Ezen, egyik leggyakoribb SQL utasítás hatására a Microsoft Jet adatbázis-szerkesztő az adatbázisból az információkat rekordhalmazként adja vissza. Az utasítás az adatbázis adatait nem módosítja. Az utasítás argumentumai a következők:

- A feltételes *állítás* argumentum a visszaadandó rekordok számát korlátozó következő záradékok egyike lehet: **ALL**, **DISTINCT**, **DISTINCTROW** vagy **TOP**.
- A * az adott tábla vagy táblák minden mezőjét kijelöltté teszi.
- A *tábla* argumentum a kijelölt rekordokat tartalmazó mezők táblájának neve.
- A *mező* argumentumok azon mezőnevek listáját adják, amely mezőkről adatokat akarunk kiolvasni. A visszaadott adatok a mezőnevek listájának sorrendjét követik.
- A *hivatkozási_név* argumentumok a tábla eredeti oszlopnevei helyett használandó oszlopfejlécek neveit tartalmazzák.
- A *táblakifejezés* argumentumok a kiolvasandó adatot tartalmazó táblák nevei.
- Ha a *táblakifejezés* argumentum által kijelölt tábla nem az aktuális adatbázisban található, annak nevét a *külső_adatbázis* argumentum tartalmazza.

A **SELECT** végrehajtásakor a Jet adatbázis-szerkesztő megkeresi a *táblakifejezés* argumentumban adott táblát vagy táblákat, abban a *mező1*, *mező2*, ... argumentum szerinti oszlopot és sort, majd az eredményt a feltételes szabálynak megfelelően rendezi és/vagy csoportosítja.

Mint az a szintaxisból látható, a **SELECT** utasítás legegyszerűbb, a feltételes argumentumokat nem tartalmazó alakja a **SELECT** *mező* **FROM** *tábla*, ahol a *mező* a {* | *tábla.** | [*tábla.*]*mező1* [, [*tábla.*]*mező2* [, ...]]} kifejezés alapján a következő alakok valamelyikeként határozható meg:

- a * az összes mezőt,
- a *tábla.** a több táblán is megtalálható mezők közül a megadott nevű tábla összes mezőjét,
- a *tábla.mező1*, *tábla.mező2*, ... pedig a több táblán is megtalálható mezők közül egy adott tábla adott, *mező1*, *mező2*, ... mezőit teszi kijelöltté.

Egy **Recordset** objektum létrehozásakor a Microsoft Jet adatbázis-szerkesztő a tábla mezőnevét használja **Field** objektumnévként. Ha másik vagy mezőt létrehozó kifejezést nem magábanfoglaló mezőnevet kívánunk használni, az utasítást a feltételes **AS** *hivatkozási_név1* záradékkal kell használni. Az olyan globális függvények vagy lekérdezések esetén, amelyek bizonytalan vagy kettős **Field** objektumneveket adnak vissza, az **AS** záradékkal adhatunk a **Field** számára másik nevet.

A **SELECT** utasítás feltételes **WHERE**, **GROUP BY**, **HAVING**, **ORDER BY** és **WITH OWNERACCESS OPTION** záradékokkal is kiegészíthető, így további korlátozásokat és adatrendezési módot előírva.

A **SELECT** utasítás egy **QueryDef** objektum SQL tulajdonságában, egy **Data** ve-

zérlőelem **RecordSource** tulajdonságában vagy az **OpenRecordset** metódus argumentumaként használható.

A Visual Basic programban az alkalmazás által adott adatok kiválasztására, rendezésére vagy szűrésére a helyi változók egy utasításba történő összefűzésével is létrehozható **SELECT** utasítás.

SELECT...INTO

Szintaxis:

SELECT *mező1* [, *mező2* [, ...]] **INTO** *újtábla* [**IN** *külső_adatbázis*]
FROM *forrás*

Ez az SQL utasítás egy táblakészítő lekérdezést hajt végre. Az utasítás argumentumai a következők:

- A *mező* argumentumok azon mezőnevek listáját adják, amelyeket az új táblába kell másolni. A másolás a mezőnevek listájának sorrendjét követi.
- Az *újtábla* argumentum a létrehozandó tábla névadási konvencióknak megfelelő neve. Már létező tábla nevének megadása hibát okoz. Az új tábla számára elsődleges kulcsot lehet definiálni. A tábla létrehozásakor annak mezői öröklik a lekérdezés alatti táblák minden egyes adattípusát és mezőméretetét.
- A feltételes *külső_adatbázis* argumentum a külső adatbázis elérési útját határozza meg.
- A *forrás* argumentum azon létező tábla vagy lekérdezés neve, amelyből a rekordok kijelölése történik.

Mint az a szintaxisból látszik, a **SELECT** utasítás *forrás* argumentumában adott táblából a *mező* argumentumokban szereplő mezőket az *újtábla* nevű, újonan létrehozott táblába másolja. Az *újtábla* esetleg egy *külső_adatbázis* argumentumban adott külső adatbázisban is lehet. A táblakészítő lekérdezés alkalmazási területe a rekordok archiválása, táblákról biztonsági másolat vagy másik adatbázisba exportáláshoz történő másolat készítése, valamint a meghatározott időközönként megjelenítendő jelentések alapjakénti használat.

Meglévő táblához adatok hozzáadására az **INSERT INTO** utasítást kell használni.

A táblakészítő lekérdezés végrehajtása előtt a kiválasztott rekordok egy azonos kiválasztási feltételű **SELECT** választó lekérdezés eredménye alapján ellenőrízhetők.

TRANSFORM

Szintaxis:

TRANSFORM *csoportfüggvény*
kiválasztási_utasítás

PIVOT *kimutatásmező* [**IN** (*érték1* [, *érték2* [, ...]])]

Ez az SQL utasítás egy kereszttáblás lekérdezést valósít meg. Az utasítás lehetséges argumentumai a következők:

- A *csoportfüggvény* argumentumban a kiválasztott adatokon végrehajtandó csoportfüggvényt adja meg.
- A *kiválasztási_utasítás* argumentum egy **SELECT** utasítás.
- A *kimutatásmező* argumentum egy mező vagy kifejezés, amelyet a lekérdezések eredményhalmazában az oszlopfej létrehozására kívánunk használni.
- A feltételes *érték* argumentumokban az oszlopfej létrehozására használt rögzített értékek találhatók, amelyek korlátozzák a *kimutatásmező* argumentum hatását. Adathoz nem tartozó érték megadásával további oszlop hozható létre.

Az adatok kereszttáblás lekérdezéssel történő összegzése során a meghatározott mezőkből vagy oszlopfej-kifejezésekből választott adatokat az eredeti lekérdezésnél sokkal tömörebb formában lehet szemléltetni.

A **TRANSFORM** utasításnak az SQL karakterláncban az első helyen, a sorfejként adott mezőket kijelölő **SELECT** utasítást és a sorcsoportokat kijelölő **GROUP BY** záradékot megelőzően kell állnia. Feltételesen egyéb — például **WHERE** — záradék is használható további kiválasztási és rendezési feltételek megadására.

UPDATE

Szintaxis:

UPDATE *tábla*
SET *újérték*
WHERE *feltétel*

Ez az SQL utasítás egy frissítő lekérdezést hajt végre. Az utasítás lehetséges argumentumai a következők:

- A *tábla* argumentumban annak a táblának a nevét kell megadni, amelynek adatait módosítani kívánjuk.
- A **SET** *újérték* egy kifejezés, amely a frissítendő rekordok adott mezőjébe kerülő értéket határozza meg.
- A frissítendő rekordok kiválasztását a **WHERE** *feltétel* kifejezés írja elő.

Az **UPDATE** utasítás elsősorban akkor használatos, ha egyidőben több rekordot kívánunk módosítani vagy ha a rekordok többszörös táblákon találhatók.

Mivel az **UPDATE** nem állít elő eredményt, a frissítő lekérdezés végrehajtása előtt a módosításra kerülő rekordok egy azonos kiválasztási feltételű **SELECT** választó lekérdezés eredménye alapján ellenőrízhetők.

Állítások

ALL

Szintaxis:

SELECT ALL
FROM *tábla*

Ez az SQL állítás a lekérdezésekkel kiválasztott rekordokat határozza meg: a Microsoft Jet adatbázis-szerkesztő az SQL utasításban szereplő feltételeknek megfelelő összes rekordot kiválasztja a **FROM** záradékkal kijelölt *tábla* rekordjai közül. Ha más állítást nem használunk, alapértelmezetten érvényes.

DISTINCT

Szintaxis:

SELECT DISTINCT
FROM *tábla*

Ez az SQL állítás a lekérdezésekkel kiválasztott rekordokat határozza meg: a **FROM** záradékkal kijelölt *tábla* rekordjainak kijelölt mezőiben az azonos adatokat tartalmazó rekordokat figyelmen kívül hagyja. A **SELECT** utasításban listázott minden egyes mező értékének egyedisége esetén lekérdezés eredményében beleértett.

Ha a **SELECT** záradék egynél több rekordot tartalmaz, az összes mezőértékek kombinációjának egyedinek kell lennie ahhoz, hogy egy adott rekord az eredménybe kerüljön. A **DISTINCT** állítást használó lekérdezés eredménye nem frissíthető és nem adja vissza a többi felhasználó későbbi módosításait.

DISTINCTROW

Szintaxis:

SELECT DISTINCTROW
FROM *tábla*

Ez az SQL állítás a lekérdezésekkel kiválasztott rekordokat határozza meg: a **FROM** záradékkal kijelölt *tábla* rekordjainak kijelölt mezőiben az azonos adatokat tartalmazó teljes rekordokat figyelmen kívül hagyja. Az állítás csak akkor hatásos, ha a lekérdezésben használt táblákból több, de nem az összes mezőt kiválasztjuk. A **DISTINCTROW** figyelmen kívül marad, ha a lekérdezés csak egy táblát tartalmaz vagy minden táblából mezőket olvasunk ki.

TOP

Szintaxis:

SELECT TOP *n* [**PERCENT**]
FROM *tábla*
ORDER BY *mező1*, *mező2*
WHERE *feltétel*

Ez az SQL állítás a lekérdezésekkel kiválasztott rekordokat határozza meg: a **FROM** záradékkal kijelölt *tábla* rekordjainak kijelölt mezőiben *n* számú rekordot ad vissza egy **ORDER BY** záradék által meghatározott mezőtartomány tetejéről vagy aljáról. Az *n* előjel nélküli **Integer** érték. A kívánt tábla nevét a **WHERE** záradékkal jelölhetjük ki. A feltételes **PERCENT** kulcsszóval a rekordok *n* százaléka adható vissza a vissza a meghatározott tartomány tetejéről vagy aljáról.

A **TOP** állítás nem választ azonos értékek között, és nincs hatása a lekérdezés frissíthetőségére.

Deklarációk

PARAMETERS

Szintaxis: **PARAMETERS** *név adattípus* [, *név adattípus* [, ...]]

Ebben a deklarációban egy paraméteres lekérdezésben adhatjuk meg minden egyes paraméter nevét és adattípusát. A deklaráció argumentumai a következők:

- A *név* argumentumok a paraméterek nevei. A **Paraméter** objektum **Name** tulajdonságának beállításával az adott paramétert ezzel a névvel lehet azonosítani a **Parameters** objektumgyűjteményben. A név karakterláncként használható, amely egy párbeszédpanelen jelenik meg a lekérdezés alkalmazásunk általi futtatása alatt. A szóközt vagy írásjeleket tartalmazó szöveget szögletes zárójelek között kell megadni.
- Az *adattípus* argumentumok a paraméter adattípusa.

A rendszeresen futtatott lekérdezések számára a **PARAMETERS** deklaráció használatával egy paraméteres lekérdezést lehet létrehozni, ezzel segítve a lekérdezési feltételek változtatásának automatizálását. Egy paraméteres lekérdezéssal a programkódnak egy lekérdezés minden egyes futtatásakor paramétereket kell szolgáltatnia.

A **PARAMETERS** deklaráció feltételes, azonban használatakor — a **SELECT** utasítást is beleértve — minden más utasítást meg kell előznie.

Ha a deklaráció több paramétert is tartalmaz, azokat egymástól vesszővel elválasztva kell felsorolni.

A **WHERE** és **HAVING** záradékokban a *név* argumentum használható, azonban az *adattípus* nem.

WITH OWNERACCESS OPTION

Szintaxis: *sqlutasítás* **WITH OWNERACCESS OPTION**

Ezt a feltételes deklarációt többfelhasználós környezetben egy biztonsági munkacsoporttal használva, egy lekérdezéssel a **WITH OWNERACCESS OPTION** a lekérdezést futtató felhasználónak a lekérdezésben lévő tábla használatához az arra vonatkozó jogosultsága helyett a lekérdezés tulajdonosként érvényessel azonos jogosultságot ad. A deklaráció argumentuma a következő:

- Az *sqlutasítás* argumentum az SQL utasítások listája.

Ha a felhasználó számára valamilyen okból tiltott egy tábla létrehozása vagy táblához lekérdezés hozzáadása, a **WITH OWNERACCESS OPTION** deklaráció lehetővé teszi számára a táblakészítő vagy hozzáfűző lekérdezés futtatása. A használat feltétele az adatbázissal kapcsolatos SYSTEM.MDW fájlhoz történő hozzáférés.

Ha a munkacsoport biztonsági beállításokat és felhasználói beállításokat érvényben ajarjuk tartani, a **WITH OWNERACCESS OPTION** deklaráció nem használható.

Műveletek

INNER JOIN

Szintaxis:

FROM *tábla1* **INNER JOIN** *tábla2*
ON *tábla1.mező1 összehasonlítási_operátor tábla2.mező2*

Ez a művelet mindannyiszor egyesíti két tábla mezőit, ha egy közös mezőben egymással egyező értékek találhatók. A művelet argumentumai a következők:

- A *tábla* argumentumok az egyesítendő rekordokat tartalmazó táblák nevei.
- A *mező* argumentumok az egyesítendő mezők nevei. Ha a mezők nem numerikusak, ugyanazon adattípusúaknak kell lenniük és ugyanazon fajta adatot kell tartalmazniuk, azonban nem szükséges azonos nevűeknek lenniük. A numerikus mezők akkor is egyesíthetők, ha nem azonos adattípusúak.
- Az *összehasonlítási_operátor* argumentum bármelyik ("=", "<", ">", "<=",">=", "<>") összehasonlítási operátor lehet.

Az **INNER JOIN** művelet bármely **FROM** záradékban használható és azonossági vagy másként belső egyesítést végez. Ez az azonossági egyesítés a legáltalánosabb egye-

sítési mód, amely két tábla rekordjait akkor egyesíti, ha mindkét tábla azonos mezőjében azonos értékek találhatók.

A Memo vagy OLE Object adatokat tartalmazó mezők egyesítése hibát okoz.

Egy **INNER JOIN** műveletbe **LEFT JOIN** és **RIGHT JOIN** műveletek is ágyazhatók, de ez fordítva nem érvényes.

LEFT JOIN

Szintaxis:

FROM *tábla1* **LEFT JOIN** *tábla2*
ON *tábla1.mező1 összehasonlítási_operátor tábla2.mező2*

Ez a művelet mindannyiszor egyesíti két tábla mezőit, ha egy közös mezőben egymással egyező értékek találhatók. A művelet argumentumai a következők:

- A *tábla* argumentumok az egyesítendő rekordokat tartalmazó táblák nevei.
- A *mező* argumentumok az egyesítendő mezők nevei. Ha a mezők nem numerikusak, ugyanazon adattípusúaknak kell lenniük és ugyanazon fajta adatot kell tartalmazniuk, azonban nem szükséges azonos nevűeknek lenniük. A numerikus mezők akkor is egyesíthetők, ha nem azonos adattípusúak.
- Az *összehasonlítási_operátor* argumentum bármelyik ("=", "<", ">", "<=",">=", "<>") összehasonlítási operátor lehet.

A **LEFT JOIN** művelet bármely **FROM** záradékban használható és a két tábla közül az első (bal oldali) összes rekordját magában foglaló külső egyesítését végzi. Az egyesítés akkor is megtörténik, ha a második (jobb oldali) tábla rekordjai nem illenek össze.

Olyan lekérdezés létrehozása, amely csak olyan rekordokat tartalmaz, amelyek egyesített mezőiben az adatok azonosak, az **INNER JOIN** műveletet kell használni.

Egy **INNER JOIN** műveletbe **LEFT JOIN** és **RIGHT JOIN** műveletek is ágyazhatók, de ez fordítva nem érvényes.

A Memo vagy OLE Object adatokat tartalmazó mezők egyesítése hibát okoz.

RIGHT JOIN

Szintaxis:

FROM *tábla1* **RIGHT JOIN** *tábla2*
ON *tábla1.mező1 összehasonlítási_operátor tábla2.mező2*

Ez a művelet mindannyiszor egyesíti két tábla mezőit, ha egy közös mezőben egymással egyező értékek találhatók. A **RIGHT JOIN** művelet bármely **FROM** záradékban használható és a két tábla közül a második (jobb oldali) összes rekordját magában fog-

laló külső egyesítését végzi.

A műveletre vonatkozó tudnivalók értelemszerűen azonosak a **LEFT JOIN** műveletnél leírtakkal.

UNION

Szintaxis:

[**TABLE**] *lekérdezés1* **UNION** [**ALL**]
[**TABLE**] *lekérdezés2* [**UNION** [**ALL**] [**TABLE**] *lekérdezés-n* [...]]

Ez a művelet egy egyesítő lekérdezést hoz létre, két vagy több független lekérdezés vagy tábla eredményének egyesítésével. A művelet argumentuma a következő:

- A *lekérdezés* argumentumok egy **SELECT** utasítás, egy tárolt lekérdezés neve vagy egy tárolt tábla neve, amelyet a **TABLE** kulcsszó előz meg. Minden egyes *lekérdezés* argumentumban egy **GROUP BY** és/vagy **HAVING** záradék használatával a visszaadott adatok csoportosíthatók, a *lekérdezés* után elhelyezett **ORDER BY** záradékkal pedig adott sorrendben jeleníthetők meg.

Egyetlen **UNION** műveletben két vagy több lekérdezés eredményét, táblákat és **SELECT** utasításokat lehet kombinálni. Alapértelmezetten a **UNION** művelet nem ad vissza többszörös rekordokat, azonban az **ALL** állítás használatával nem csak minden rekord visszaadása írható elő, de a lekérdezés is gyorsabbá válik.

Egy **UNION** művelet mindegyik lekérdezése csak azonos számú mezőt érinthet, azonban az egyes mezőknek nem kell ugyanazon méretűnek vagy adattípusúnak lennie.

Operátorok

Between...And

Szintaxis: *kifejezés* [**Not**] **Between** *érték1* **And** *érték2*

Ez az SQL operátor azt határozza meg, hogy egy kifejezés értéke meghatározott értéktartományba esik-e. Az operátor csak SQL kifejezésen belül használható. Az operátor szintaxisa a következő részeket tartalmazza:

- A *kifejezés* a kiértékelendő adatot tartalmazó mezőt azonosító kifejezés.
- Az *érték1* és *érték2* részek a *kifejezés* által kijelölt mező kiértékeléséhez használandó kifejezések.

Ha a *kifejezés* értéke az *érték1* és *érték2* közötti, az operátor **True** értéket ad vissza, egyébként az eredmény **False** lesz. A feltételes **Not** operátor megfordítja a feltételeket, azaz a *kifejezés* adott tartományon kívül eső értéke esetén kapunk **True** eredményt.

In

Szintaxis: *kifejezés* [**Not**] **In** (*érték1*, *érték2*, ...)

Ez az SQL operátor azt határozza meg, hogy egy kifejezés értéke azonos-e egy lista valamelyik értékével. Az operátor szintaxisa a következő részeket tartalmazza:

- A *kifejezés* a kiértékelendő adatot tartalmazó mezőt azonosító kifejezés.
- Az *érték1* és *érték2* részek a *kifejezés* által kijelölt mező kiértékeléséhez használandó kifejezések.

Ha a *kifejezés* értéke megtalálható az *érték1*, *érték2*, ... értéklistában, az operátor **True** értéket ad vissza, egyébként az eredmény **False** lesz. A feltételes **Not** logikai operátor megfordítja a feltételeket, vagyis akkor kapunk **True** eredményt, ha a *kifejezés* értéke nem található az értéklistában.

Like

Szintaxis: *kifejezés* **Like** "*minta*"

Ez az SQL operátor egy karakteres kifejezést hasonlít össze egy SQL kifejezésbeli mintával. Az operátor szintaxisa a következő részeket tartalmazza:

- A *kifejezés* egy **WHERE** záradékban használt SQL kifejezés.
- A *minta* az a karakteres kifejezés, amellyel a *kifejezés* részben adott SQL kifejezést össze kell hasonlítani.

A **Like** operátort egy mezőben adott mintának megfelelő mezők keresésére lehet használni. A minta nem csak teljes alakú, hanem a rendszer által ismert helyettesítő, un. wildcard karaktereket (például ? vagy *) is tartalmazó kifejezés lehet.

Egy SQL lekérdezésben például a `Like "P[ A-E ]###"` eredménye az összes P betűvel kezdődő, másodikként A, B, C, D vagy E betűt tartalmazó, és három betű hosszúságú adat lesz.

Záradékok

CONSTRAINT

Szintaxis 1 (index):

CONSTRAINT *név* {**PRIMARY KEY** |
UNIQUE |
REFERENCES *idegentábla* [(*idegenmező1*, *idegenmező2*)]}

Szintaxis 2 (multiindex):

CONSTRAINT *név*
{**PRIMARY KEY** *elsődleges1*, [, *elsődleges2* [, ...]]) |
UNIQUE (*egyedi1*, [, *egyedi2*, [, ...]]) |
FOREIGN KEY (*hivatkozás1* [, *hivatkozás2* [, ...]])
REFERENCES *idegentábla* [(*idegenmező1* [, *idegenmező2* [, ...]])]}

Ez a záradék az indexhez hasonló, bár táblák közötti kapcsolat meghatározására is alkalmazható. A **CONSTRAINT** záradékot a táblát törlő **ALTER TABLE** vagy a táblát létrehozó **CREATE TABLE** utasításban használhatjuk. Mint látható, a záradék az egymezős és a többmezős esetekre eltérő szintaxis szerint használható. A záradék argumentumai a következők:

- A *név* argumentum a létrehozandó kényszer neve.
- Az *elsődleges* argumentumok az elsődleges kulcs által kiválasztott mezők nevei.
- Az *egyedi* argumentumok az egyedi kulcs által kiválasztott mezők nevei.
- A *hivatkozás* argumentumok egy másik tábla mezőire hivatkozó idegen kulcsmezők nevie.
- Az *idegentábla* argumentum aannak az idegen táblának a neve, amely az *idegenmező* argumentumban adott mezőket tartalmazza.
- Az *idegenmező* argumentumok az *idegentábla* által kijelölt táblában a *hivatkozás* argumentummal megadott mezők nevei. Elhagyható, ha a hivatkozott mező az *idegentábla* elsődleges kulcsa.

Az egymezős szintaxis egy **ALTER TABLE** vagy **CREATE TABLE** utasításban a mező adattípusának megadása után használandó.

A több mezőre vonatkozó multiindexes szintaxis akkor használható, ha a **CONSTRAINT** záradék az **ALTER TABLE** vagy **CREATE TABLE** utasításban szereplő meződefiniáló záradékon kívül szerepel.

A **CONSTRAINT** használatakor egy mezőt a következő kényszertípus szerint lehet kiválasztani:

- A **UNIQUE** kulcsszót használva egy mezőt egyedi mezőként választhatunk ki, ami azt jelenti, hogy a táblában nem lehet két, ugyanazon értékű érték, azaz minden táblának egyedinek kell lennie. Ha egy multiindex van egyedi kulcsként kiválasztva, az index minden mezőjének kombinált értékeinek még akkor is egyedieknek kell lenniük, ha egy vagy több mezőben egy vagy több rekord értéke azonos.
- A **PRIMARY KEY** kulcsszó használatával egy táblában egy mező vagy mezőhalmaz elsődleges kulcsként választható ki. Az elsődleges kulcsban minden értéknek egyedinek kell lennie, nem lehet **Null**, továbbá egy táblának csak egy elsődleges kulcsa lehet. A **PRIMARY KEY** kulcsszónak elsődleges kulccsal már

rendelkező táblára történő használata hibát okoz.

- A **FOREIGN KEY** kulcsszó használatával egy mező idegen kulcsként választható ki. Ha az idegen tábla elsődleges kulcsa egynél több mezőt tartalmaz, a multiindexes alakot kell használni, felsorolva az összes hivatkozó mezőt, az idegen tábla nevét, és az idegen táblabeli hivatkozott mezőket az idegen táblabeli sorrendjüknek megfelelően. Ha a hivatkozott mező vagy mezők az idegen tábla elsődleges mezői, nem szükségs a hivatkozott mezőket felsorolni, a Jet az elsődleges kulcsot alapértelmezetten hivatkozott mezőknek tekinti.

FROM

Szintaxis:

SELECT *mezőlista*
FROM *táblakifejezés* [**IN** *külső_adatbázis*]

Ez a kötelező záradék a **SELECT** utasítást követően a listázott mezőket tartalmazó táblákat vagy lekérdezéseket határozza meg. A záradék argumentumai a következők:

- A *mezőlista* argumentum bármely mezőnév-helyettesítőkkel, SQL összegző függvényekkel, kiválasztási állításokkal (**ALL**, **DISTINCT**, **DISTINCTROW** vagy **TOP**) vagy más **SELECT** utasítás beállításokkal lekérdezhető mező vagy mezők neve.
- A *táblakifejezés* argumentum egy vagy több, a lekérdezendő adatokat tartalmazó táblát megadó kifejezés: egy tábla neve, egy tárolt lekérdezés neve vagy ezekből az **INNER JOIN**, **LEFT JOIN** vagy **RIGHT JOIN** művelettel kapott keverék. A táblák nevének sorrendje közömbös.
- A *külső_adatbázis* argumentum a *táblakifejezés* összes tábláit tartalmazó külső adatbázis teljes elérési útja.

A jellemzők javítására és az egyszerűbb használat érdekében célszerű a külső adatbázisból történő adatekérdezéséhez az **IN** záradék helyett csatolt táblát használni.

GROUP BY

Szintaxis:

SELECT *mezőlista*
FROM *táblakifejezés*
WHERE *feltétel*
GROUP BY *csoportmezőlista*

Ez a feltételes záradék a megadott mezőlista azonos értékű rekordjait egyetlen rekordba egyesíti. Egy SQL összegző függvényben, például **Sum( )** vagy **Count( )** függ-

vényben történő használata esetén minden egyes rekordhoz egy összegzett érték jön létre. A záradék argumentumai a következők:

- A *mezőlista* argumentum bármely mezőnév-helyettesítőkkel, SQL összegző függvényekkel, kiválasztási állításokkal (**ALL**, **DISTINCT**, **DISTINCTROW** vagy **TOP**) vagy más **SELECT** utasítás beállításokkal lekérdezhető mező vagy mezők neve. Ha SQL összegző függvény nem használt, az összefoglalt értékek figyelmen kívül maradnak.
- A *táblakifejezés* argumentum egy vagy több, a lekérdezendő adatokat tartalmazó táblát megadó kifejezés: egy tábla neve, egy tárolt lekérdezés neve vagy ezekből az **INNER JOIN**, **LEFT JOIN** vagy **RIGHT JOIN** művelettel kapott keverék. A táblák nevének sorrendje közömbös.
- A *feltétel* argumentum a kiválasztási feltételt tartalmazza. Ha az utasítás egy **WHERE** záradékot tartalmaz, a Microsoft Jet adatbázis-szerkesztő a **WHERE** rekordokra alkalmazása után az értékeket csoportosítja.
- A *csoportmezőlista* a csoport rekordokhoz használt, legfeljebb tíz mező neve. A mezőnevek sorrendje a csoportosítási szinteket a legmagasabbtól a legalacsonyabb csoportosítási szintig határozza meg.

A **GROUP BY** mezőiben a **Null** értékek nem maradnak figyelmen kívül, azok kis csoportosításra kerülnek, azonban semelyik SQL összegző függvényben sem értékelődnek ki.

A csoportosításból kizárandó sorokhoz a **WHERE** záradék, a csoportosítás után a rekordok szűrésére pedig a **HAVING** záradék használható.

A **GROUP BY** mezőlistában egy mező — hacsak nem Memo vagy OLE Object adatot tartalmaz — még akkor is bármely tábla mezőjére hivatkozhat a **FROM** záradékban, ha a mező nem található a **SELECT** utasításban, feltéve hogy a **SELECT** utasítás legalább egy összegző függvényt tartalmaz. A Jet adatbázis-szerkesztő nem csoportosít Memo vagy OLE Object mezőket.

A **SELECT** *mezőlista* mindegyik mezőjének vagy a **GROUP BY** záradékban, vagy egy SQL összegző függvény argumentumában kell szerepelnie.

A szóközt vagy írásjeket tartalmazó mezőneveket szögletes zárójelben kell megadni.

HAVING

Szintaxis:

SELECT *mezőlista*
FROM *táblakifejezés*
WHERE *feltétel*
GROUP BY *csoportmezőlista*
HAVING *csoportfeltétel*

Ez a feltételes záradék egy **GROUP BY** záradékkal használt **SELECT** utasításban megjelenő csoportosított rekordokat határozza meg. Amint a **GROUP BY** a rekordokat egyesíti, a **HAVING** ezek közül a feltételének megfelelő minden rekordot megjeleníti. A záradék argumentumai a következők:

- A *mezőlista* argumentum bármely mezőnév-helyettesítőkkel, SQL összegző függvényekkel, kiválasztási állításokkal (**ALL**, **DISTINCT**, **DISTINCTROW** vagy **TOP**) vagy más **SELECT** utasítás beállításokkal lekérdezhető mező vagy mezők neve. Ha SQL összegző függvény nem használt, az összefoglalt értékek figyelmen kívül maradnak.
- A *táblakifejezés* argumentum egy vagy több, a lekérdezendő adatokat tartalmazó táblát megadó kifejezés: egy tábla neve, egy tárolt lekérdezés neve vagy ezekből az **INNER JOIN**, **LEFT JOIN** vagy **RIGHT JOIN** művelettel kapott keverék. A táblák nevének sorrendje közömbös.
- A *feltétel* argumentum a kiválasztási feltételt tartalmazza. Ha az utasítás egy **WHERE** záradékot tartalmaz, a Microsoft Jet adatbázis-szerkesztő a **WHERE** rekordokra alkalmazása után az értékeket csoportosítja.
- A *csoportmezőlista* argumentum a csoport rekordokhoz használt, legfeljebb tíz mező neve. A mezőnevek sorrendje a csoportosítási szinteket a legmagasabbtól a legalacsonyabb csoportosítási szintig határozza meg.
- A *csoportfeltétel* argumentum a csoportosított rekordok közül a megjelenítendőket meghatározó kifejezés.

A **HAVING** hasonló a **WHERE** záradékhoz, amely ugyancsak kijelölt rekordokat határoz meg. Amint a rekordok a **GROUP BY** záradéknak megfelelően csoportosítva vannak, közülük a **HAVING** záradék határozza meg a megjelenítendőket.

A **HAVING** záradék legfeljebb 40, egymáshoz logikai (például **And** vagy **Or**) operátorokkal öszekapcsolt kifejezést tartalmazhat.

IN

Szintaxis 1 (céltábla azonosításhoz):

[SELECT | INSERT] INTO *cél* **IN**
{ *elérési_út* | ["*elérési út*" "*típus*"] | ["" [*típus* ; DATABASE = *elérési_út*]]}

Szintaxis 2 (forrástábla azonosításhoz):

[FROM *táblakifejezés* **IN**
{ *elérési_út* | ["*elérési út*" "*típus*"] | ["" [*típus* ; DATABASE = *elérési_út*]]}

Ez a záradék táblákat határoz meg egy külső, például dBASE, Paradox vagy külső Jet adatbázisban, amelyhez a Microsoft Jet adatbázis-szerkesztő csatlakozhat. A záradék argumentumai a következők:

- A *cél* argumentum az adatokat tartalmazó külső adatbázis neve.
- A *táblakifejezés* argumentum egy vagy több, a lekérdezendő adatokat tartalmazó táblát megadó kifejezés: egy tábla neve, egy tárolt lekérdezés neve vagy ezekből az **INNER JOIN**, **LEFT JOIN** vagy **RIGHT JOIN** művelettel kapott keverék. A táblák nevének sorrendje közömbös.
- Az *elérési_út* argumentum a táblát tartalmazó könyvtár vagy fájl teljes elérési útja.
- A *típus* argumentum a tábla létrehozásához használt adatbázistípus neve, ha az adatbázis nem Jet adatbázis, hanem pldául dBASEIII, dBASE IV, Paradox 3.x, Paradox 4.x vagy Btrieve.

Az **IN** záradékot egyidőben csak egy külső adatbázishoz kapcsolódáshoz lehet használni.

Néhány esetben az *elérési_út* argumentum az adatbázisfájlokat tartalmazó könyvtárra hivatkozik. A tábla fájlneve a *cél* vagy *táblakifejezés* argumentumokból származtatható.

A nem Jet adatbázis megadásához a nevéhez egy pontosvesszőt kell adni és az egyszerű vagy kettős idézőjelbe kell helyezni, például 'dBASEIV;' vagy "dBASEIV;" alakban.

A külső adatbázis a **DATABASE** kulcsszó használatával is kijelölhető, például a következő alakban: `... FROM Table IN "" [ dBASE IV; DATBASE = C:\DBASE\DATA\RENDELES; ];`

A jellemzők javítására és az egyszerűbb használat érdekében célszerű a külső adatbázisból történő adatlekérdezéséhez az **IN** záradék helyett csatolt táblát használni.

Az **IN** kulcsszó egy kifejezésben összehasonlítási operátorként is használható.

ORDER BY

Szintaxis:

SELECT *mezőlista*
FROM *táblakifejezés*
WHERE *feltétel*
ORDER BY *mező1* [**ASC** | **DESC**] [, *mező2* [**ASC** | **DESC**] [, ...]]

Ez az SQL kifejezésekben általában legutolsóként használt feltételes záradék egy adott mező vagy mezők lekérdezésének eredményeként kapott rekordokat növekvő vagy csökkenő sorrendbe rendezi. A záradék argumentumai a következők:

- A *mezőlista* argumentum bármely mezőnév-helyettesítőkkel, SQL összegző függvényekkel, kiválasztási állításokkal (**ALL**, **DISTINCT**, **DISTINCTROW** vagy **TOP**) vagy más **SELECT** utasítás beállításokkal lekérdezhető mező vagy mezők neve. Ha SQL összegző függvény nem használt, az összefoglalt értékek

figyelmen kívül maradnak.

- A *táblakifejezés* argumentum egy vagy több, a lekérdezendő adatokat tartalmazó táblát megadó kifejezés: egy tábla neve, egy tárolt lekérdezés neve vagy ezekből az **INNER JOIN**, **LEFT JOIN** vagy **RIGHT JOIN** művelettel kapott keverék. A táblák nevének sorrendje közömbös.
- A *feltétel* argumentum a kiválasztási feltételt tartalmazza. Ha az utasítás egy **WHERE** záradékot tartalmaz, a Microsoft Jet adatbázis-szerkesztő a **WHERE** rekordokra alkalmazása után az értékeket csoportosítja.
- A *mező* argumentumok azon mezők nevei, amelyekben a rekordokat rendezni kívánjuk. csoport rekordokhoz használt, legfeljebb tíz mező neve. A mezőnevek sorrendje a csoportosítási szinteket a legmagasabbtól a legalacsonyabb csoportosítási szintig határozza meg.
- A *csoportfeltétel* argumentum a csoportosított rekordok közül a megjelenítendőket meghatározó kifejezés.

Az **ORDER BY** záradék nélkül az adatok rendezettlenül jelennek meg.

A záradékban a rendezést az **ASC** kulcsszó növekvő (A-Z vagy 0-9), a **DESC** kulcsszó pedig csökkenő (Z-A vagy 9-0) irányba állítja be. Az alapértelmezett rendezési irány növekvő, így az **ASC** használata nem kötelező.

Az **ORDER BY** záradékban Memo vagy OLE Object mezők megadása hibát okoz, a Jet adatbázis-szerkesztő nem rendezi az ilyen típusú mezőket.

PROCEDURE

Szintaxis: **PROCEDURE** *név* [*paraméter1*, *adattípus* [, *paraméter2*, *adattípus* [, ...]] .

Ebben a záradékban nevek és feltételes paraméterek adhatók meg egy lekérdezéshez. A záradék argumentumai a következők:

- A *név* argumentum az eljárás standard névadási konvenciónak megfelelő alakú neve.
- A *paraméter* argumentumok egy vagy több mezőnevet vagy paramétert határoznak meg.
- Az *adattípus* argumentum egyike a 13 Microsoft Jet SQL adattípusainak vagy azok szinonímáinak.

Egy SQL eljárás egy **PROCEDURE** záradékot, egy feltételes paraméterdefiníciós listát és egyetlen, kötelező SQL — például **SELECT** vagy **UPDATE** — utasítást tartalmaz.

Ha a záradék több meződefiníciót (*paraméter* — *adattípus* párt) is tartalmaz, azokat egymástól vesszővel elválasztva kell felsorolni.

WHERE

Szintaxis:

SELECT *mezőlista*
FROM *táblakifejezés*
WHERE *feltétel*

Ez a feltételes záradék azt határozza meg, meg, hogy a **FROM** záradékkban listázott táblákból mely rekordokat érintsen egy **SELECT**, **UPDATE** vagy **DELETE** utasítás. A záradék argumentumai a következők:

- A *mezőlista* argumentum bármely mezőnév-helyettesítőkkel, SQL összegző függvényekkel, kiválasztási állításokkal (**ALL**, **DISTINCT**, **DISTINCTROW** vagy **TOP**) vagy más **SELECT** utasítás beállításokkal lekérdezhető mező vagy mezők neve. Ha nem használt SQL összegző függvény, az összefoglalt értékek figyelmen kívül maradnak.
- A *táblakifejezés* argumentum egy vagy több, a lekérdezendő adatokat tartalmazó táblát megadó kifejezés: egy tábla neve, egy tárolt lekérdezés neve vagy ezekből az **INNER JOIN**, **LEFT JOIN** vagy **RIGHT JOIN** művelettel kapott keverék. A táblák nevének sorrendje közömbös.
- A *feltétel* argumentum a kiválasztási feltételt tartalmazza.

A Microsoft Jet adatbázis-szerkesztő a **WHERE** záradékban adott feltételeknnek megfelelő rekordokat választja ki. Ha nem használunk **WHERE** záradékot, a lekérdezés a tábla minden sorát visszaadja. Ha a lekérdezésben egynél több táblát adunk meg, és nincs **WHERE** vagy **JOIN** záradék, a lekérdezés a táblák kartéziusi szorzatát képezi.

A feltételes **WHERE** záradéknak a **FROM** záradék után kell következnie.

Ha nem használunk **JOIN** műveletet a többszörös táblákon végrehajtandó SQL egyesítő műveletekhez, az eredeményül kapott **Recordset** nem lesz frissíthető.

A kiválasztott rekordokat meghatározó **WHERE** a **HAVING** záradékhoz hasonló és használatával például a **GROUP BY** záradékkal csoportosítandó rekordok közül lehet a szükségteleneket eltávolítani.

A **WHERE** záradék legfeljebb 40, egymáshoz logikai (például **And** vagy **Or**) operátorokkal öszekapcsolt kifejezést tartalmazhat.

A szóközt vagy írásjeket tartalmazó mezőneveket szögletes zárójelben kell megadni.

A *feltétel* argumentum meghatározásában szereplő dátum esetén még akkor is US dátumalakot (hónap-nap-év) kell megadni, ha nem az US verzió szerinti Microsoft Jet adatbázis-szerkesztőt használjuk.

INDEX

#

#Const, 490
#If...Then...#Else, 491

A

Abs( ), 27
AbsolutePosition, 258
Action, 259
Activate, 222
ActiveControl, 259
ActiveForm, 260
Add, 137
AddItem, 138
AddNew, 138
AfterColUpdate, 222
AfterDelete, 223
AfterInsert, 223
Align, 261
Alignment, 262
ALL, 505
AllowAddNew, 262
AllowDelete, 263
AllowRowSizing, 264
AllowSizing, 264
AllowUpdate, 265
AllowZeroLength, 265
AllPermissions, 266
ALTER TABLE, 496
AppActivate, 81
Append, 139
AppendChunk, 140
AppIsRunning, 268
Archive, 268
Arrange, 141
Array( ), 27
Asc( ), 27; 28
AscB( ), 28
Atn( ), 28
AutoActivate, 271
AutoRedraw, 272
AutoShowChildren, 273
AutoSize, 274
AutoVerbMenu, 274
Avg( ), 492

B

BackColor, 275
BackStyle, 276
Beep, 82
BeginTrans, 82; 142
Between...And, 509
Binary, 119
BOF, 276
BOFAction, 278
Bold, 278
Bookmark, 279
Bookmarkable, 280
BorderColor, 281
BorderWidth, 283
BoundColumn, 284
BoundText, 285
Byte, 28

C

CacheSize, 285
CacheStart, 286
Call, 82
Cancel, 287
CancelError, 287
CancelUpdate, 143
Caption, 288
CBool( ), 28
CByte( ), 28
CCur( ), 29
CDate( ), 29
CDbl( ), 29
CellSelected, 289
CellText, 143
CellValue, 144
Change, 227
ChDir, 83
ChDrive, 83
Checked, 289
Choose( ), 30
Chr( ), 30
ChrB( ), 30
CInt( ), 30
Circle, 144
Class, 290
Clear, 145
Click, 229
Clip, 291
ClipControls, 291
CLng( ), 31
Clone, 146
Close, 83; 147
Cls, 148
Clustered, 292
Col, 293
ColAlignment, 293
ColContaining, 148
ColIndex, 294
ColIsVisible, 294
Color, 296
ColorMode, 297
ColPos, 297
Cols, 297
ColumnHeaders, 298
Columns, 298
ColWidth, 299
Command, 31
Comments, 299
CommitTrans, 84; 148
CompactDatabase, 84; 150
CompanyName, 300
ConflictTable, 300
Connect, 300
Const, 84
CONSTRAINT, 510
Container, 302
ControlBox, 303

Copies, 303
Copy, 152
CopyQueryDef, 153
Cos( ), 31
Count, 303
Count( ), 492
CreateDatabase, 31; 153
CreateDynaset, 155
CreateEmbed, 157
CreateField, 157
CreateGroup, 158
CreateIndex, 159
CreateLink, 159
CreateObject( ), 32
CreateProperty, 160
CreateQueryDef, 161
CreateSnapshot, 163
CreateTableDef, 163
CreateUser, 164
CreateWorkspace, 165
CurDir( ), 33
Currency, 29
CurrentX, 304
CurrentY, 305
CVar( ), 33
CVDate( ), 33
CVErr( ), 34
CSng( ), 32
CStr( ), 32

D

Data, 305
Database, 305
DatabaseName, 306
DataChanged, 307
DataField, 307
DataSource, 308
DataText, 309
DataUpdatable, 309
Date, 29; 34; 85
DateAdd( ), 34
DateCreated, 310
DateDiff( ), 35
DatePart( ), 36
DateSerial( ), 36
DateValue( ), 36
Dátumtagoló, 45
Day( ), 37
DblClick, 231
DDB( ), 37
Deactivate, 232
Declare, 85
Default, 310
DefaultColWidth, 313
DefaultExt, 311
DefaultPassword, 313
DefaultUser, 311
DefaultValue, 312
Def*tipus*, 87
Delete, 165
DeleteQueryDef, 167
DeleteSetting, 88
Description, 40; 313
DesignMasterID, 314
DeviceName, 315
DialogTitle, 315
Dim, 88
Dinamikus tömb, 112
Dir( ), 37
DisplayType, 315
DISTINCT, 505
DistinctCount, 316
DISTINCTROW, 505
DividerStyle, 316
Do...Loop, 90
DoEvents( ), 38
Double, 29
DoVerb, 168
Drag, 169
DragDrop, 232
DragIcon, 317
DragMode, 318
DragOver, 233
DrawMode, 319
DrawStyle, 320
DrawWidth, 320
Drive, 321
DriverName, 321
DropDown, 234
Duplex, 322

E

Edit, 170
EditMode, 322
Enabled, 323
End, 91
EndDoc, 171
Environ( ), 39
EOF, 324
EOF( ), 39
EOFAction, 325
Erase, 91
erőforrásfájl, 61
Err, 40
Error, 34; 92; 235
Error( ), 40
Exclusive, 326
Execute, 171
ExecuteSQL, 172
EXEName, 327
Exit, 92
Exp( ), 40

F

FetchVerbs, 173
Fields, 327
FieldSize, 173
FileAttr( ), 40
FileCopy, 92
FileDateTime( ), 41
FileDescription, 327
FileLen( ), 41
FileName, 328
FileNumber, 328
FileTitle, 329
FillCache, 173
FillColor, 329
FillStyle, 329
Filter, 330
FilterIndex, 331
FindFirst, 174
FindLast, 175
FindNext, 176
FindPrevious, 176
First( ), 493

FirstRow, 332
Fix( ), 41
FixedAlignment, 332
FixedCols, 333
FixedRows, 333
Flags, 333
FontBold, 338
FontCount, 338
FontItalic, 339
FontName, 339
Fonts, 340
FontSize, 340
FontStrikethru, 341
FontTransparent, 342
FontUnderline, 342
For...Next, 94
ForeColor, 343
Foreign, 343
ForeignName, 344
ForeignTable, 344
Format, 345
Format( ), 41
FreeFile( ), 48
FreeLocks, 95
FROM, 512
FromPage, 345
Function, 95
FV( ), 48

G

Get, 97
GetAllSettings( ), 49
GetAttr( ), 49
GetBookmark, 176
GetChunk, 177
GetData, 178
GetFormat, 178
GetObject( ), 49
GetRows, 179
GetSetting( ), 50
GetText, 179
GoSub...Return, 99
GotFocus, 235
GoTo, 99
GridLines, 346
GridLineWidth, 346
GROUP BY, 512

H

Handle, 346
HAVING, 513
hDC, 347
HeadBackColor, 347
HeadClick, 236
HeadFont, 348
HeadForeColor, 348
HeadLines, 349
Height, 349
HelpCommand, 350
HelpContext, 352
HelpContextID, 353
HelpFile, 354
HelpKey, 355
Hex( ), 51
hibakezelő, 106
Hidden, 355
Hide, 180
HideSelection, 355
HighLight, 356
hInstance, 356
HostName, 357
Hour( ), 51
hPal, 357
hWnd, 357

I

Icon, 358
Idle, 180
Időtagoló, 45
If...Then...Else, 100
IgnoreNulls, 358
IIf( ), 51
Image, 359
IMEStatus( ), 51
In, 510; 514
Index, 360
Inherit, 362
Inherited, 362
IniPath, 362
InitDir, 363
INNER JOIN, 507
Input#, 101
Input( ), 52
InputB( ), 52
InputBox( ), 52
Insert, 181
INSERT INTO, 500
Instancing, 364
InStr( ), 53
InStrB( ), 54
Int( ), 54
Integer, 30
IntegralHeight, 364
Interval, 365
IPmt( ), 54
IpOleObject, 382
IRR( ), 55
IsArray( ), 55
IsDate( ), 56
IsEmpty( ), 56
IsError( ), 56
IsMissing( ), 56
IsNull( ), 56
IsNumeric( ), 57
IsObject( ), 57
IsolateODBCTrans, 365
Italic, 366
Item, 182
ItemData, 367

K

Karakterláncok formázása, 47
KeepLocal, 367
KeyDown, 236
KeyPress, 238
KeyPreview, 368
KeyUp, 239
Kill, 102
KillDoc, 182
kulcsszavak, 7

L

LargeChange, 369

Last( ), 493
LastDLLError, 369
LastModified, 370
LastUpdated, 370
LBound, 370
LBound( ), 57
LCase( ), 58
Left, 370
LEFT JOIN, 508
Left( ), 58
LeftB( ), 58
LeftCol, 371
LegalCopyright, 371
LegalTradmarks, 372
Len( ), 58
LenB( ), 59
Let, 102
Like, 510
Line, 183
Line Input #, 103
LinkClose, 239
LinkError, 240
LinkExecute, 183; 241
LinkItem, 372
LinkMode, 373
LinkNotify, 241
LinkOpen, 241
LinkPoke, 184
LinkRequest, 184
LinkSend, 185
LinkTimeout, 375
LinkTopic, 375
List, 377
ListCount, 378
ListField, 378
ListFields, 185
ListIndex, 379
ListIndexes, 186
ListParameters, 186
ListTables, 186
Load, 242
LoadPicture( ), 59
LoadResData( ), 59
LoadResPicture( ), 60
LoadResString( ), 61
Loc( ), 61
Lock...Unlock, 103
Locked, 380
LockEdits, 380
LOF( ), 61
Log( ), 61
LoginTimeout, 381
LogMessages, 382
Long, 31
LostFocus, 242
LSet, 104
LTrim( ), 62

M

Major, 383
MakeReplica, 186
MatchEntry, 383
MatchWithList, 384
Max, 384
Max( ), 493
MaxButton, 385
MaxFileSize, 386
MaxLength, 386
MDIChild, 387
Mid, 104
Mid( ), 62
MidB, 105
MidB( ), 62
Min, 387
Min( ), 494
MinButton, 388
Minor, 389
Minute( ), 62
MIRR( ), 63
MiscFlags, 389
MkDir, 105
Month( ), 63
MouseDown, 243
MouseIcon, 390
MouseMove, 245
MousePointer, 390
MouseUp, 246
Move, 187
MoveFirst, 188
MoveLast, 189
MoveNext, 189
MovePrevious, 189
MsgBox( ), 63
MultiLine, 391
MultiSelect, 392

N

Name, 105; 393
Negotiate, 395
NegotiateMenus, 395
NegotiatePosition, 396
NegotiateToolbars, 396
NewIndex, 397
NewPage, 190
NewPassword, 190
NoMatch, 397
Normal, 398
Now, 65
NPer( ), 65
NPV( ), 66
Number, 399
NumberFormat, 399

O

Object, 400
ObjectAcceptFormats, 402
ObjectAcceptFormatsCount, 402
ObjectGetFormats, 402
ObjectGetFormatsCount, 403
ObjectMove, 247
ObjectVerbFlags, 403
ObjectVerbs, 403
ObjectVerbsCount, 405
Oct( ), 66
OLE Automatizmus, 50
OLEDropAllowed, 406
OLERequestPendingMsgText, 406
OLERequestPendingMsgTitle, 408
OLERequestPendingTimeout, 408
OLEServerBusyMsgText, 408
OLEServerBusyMsgTitle, 410

OLEServerBusyTimeout, 411
OLEServerRaiseError, 410
OLETimeout, 405
OLEType, 411
OleTypeAllowed, 412
On Error, 106
On...GoSub, 107
On...GoTo, 108
Open, 108
OpenDatabase, 66
OpenQueryDef, 192
OpenRecordset, 192
OpenTable, 194
Option Base, 27; 109
Option Compare, 109
Option Explicit, 110
Option Private, 110
Options, 413
ORDER BY, 515
OrdinalPosition, 414
Orientation, 415
Owner, 416

P

Page, 416
Paint, 247
PaintPicture, 194
PaperBin, 416
PaperSize, 417
PARAMETERS, 506
Parent, 419
Partition( ), 67
Password, 420
PasswordChar, 420
Paste, 195
PasteOK, 420
PasteSpecialDlg, 196
Path, 421
PathChange, 248
Pattern, 422
PatternChange, 249
PercentPosition, 422
Permissions, 423
Picture, 424
PID, 425
Pmt( ), 67
Point, 196
PopupMenu, 197
Port, 426
PPmt( ), 68
PrevInstance, 426
Primary, 426
Print, 198
Print#, 110
PrinterDefault, 427
PrintForm, 199
PrintQuality, 428
Private, 111
PROCEDURE, 516
ProductName, 428
Property Get, 113
Property Let, 114
Property Set, 116
PSet, 199
Public, 117; 429
Put, 118
PV( ), 68

Q

QBColor( ), 68
QueryTimeout, 430
QueryUnload, 249

R

Raise, 200
Random, 118
Randomize, 120
Rate( ), 69
ReadFromFile, 201
ReadOnly, 430
Rebind, 201
RecordCount, 431
RecordsAffected, 432
RecordSelectors, 432
Recordset, 433
RecordsetType, 434
RecordSource, 435
ReDim, 120
Refill, 201
Refresh, 202
RefreshLink, 203
RegisterDatabase, 121; 203
Rem, 121
Remove, 204
RemoveItem, 205
RepairDatabase, 122; 205
Replicable, 436
ReplicaID, 437
Reposition, 250
Requery, 206
Required, 437
Reset, 122
Resize, 250
Restartable, 438
Resume, 122
ReturnsRecords, 438
Revision, 439
RGB( ), 69
RIGHT JOIN, 508
Right( ), 69
RightB( ), 70
RmDir, 123
Rnd( ), 70
Rollback, 123; 207
Row, 439
RowBookmark, 208
RowColChange, 251
RowContaining, 208
RowDividerStyle, 440
RowHeight, 440
RowIsVisible, 441
RowLoaded, 252
RowPos, 441
RowResize, 252
Rows, 441
RowSource, 442
RowTop, 209
RSet, 123
RTrim( ), 70

S

SavePicture, 123
SaveSetting, 124
SaveToFile, 209

SaveToOle1File, 210
Scale, 210
ScaleHeight, 442
ScaleLeft, 443
ScaleMode, 444
ScaleTop, 444
ScaleWidth, 445
ScaleX, 211
ScaleY, 212
Scroll, 212; 253
ScrollBars, 445
Second( ), 71
Seek, 124; 212
Seek( ), 71
SelBookmarks, 446
SelChange, 254
SelCount, 446
SELECT, 501
Select Case, 125
SELECT...INTO, 503
Selected, 447
SelectedItem, 447
SelEndCol, 448
SelEndRow, 448
SelLength, 448
SelStart, 449
SelStartCol, 449
SelStartRow, 450
SelText, 450
SendKeys, 126
Set, 127
SetAttr, 128
SetData, 213
SetDataAccessOption, 128
SetDefaultWorkspace, 129
SetFocus, 214
SetText, 214
Sgn( ), 71
Shape, 451
Shell( ), 71
Shortcut, 451
Show, 215
ShowColor, 215
ShowFont, 216
ShowHelp, 216
ShowInTaskbar, 451
ShowOpen, 216
ShowPrinter, 216
ShowSave, 217
ShowWhatsThis, 217
Sin( ), 72
Single, 32
Size, 452
SizeMode, 453
SmallChange, 454
Sort, 455
Sorted, 455
Source, 456
SourceDoc, 457
SourceField, 457
SourceItem, 457
SourceTable, 458
SourceTableName, 458
Space( ), 72
Spc( ), 72
SQL, 459
Sqr( ), 73
StartMode, 459
Static, 129
StDev( ), 494
StDevP( ), 495
Stop, 130
Str( ), 73
StrComp( ), 74
StrConv( ), 74
Stretch, 460
StrikeThrough, 460
String, 32
String( ), 75
Style, 461
Sub, 130
Sum( ), 495
Switch( ), 75
SYD( ), 76
Synchronize, 217
System, 462
Számok formázása, 42

T

Tab( ), 76
TabIndex, 462
Table, 463
TabStop, 463
Tag, 464
Tan( ), 77
TaskVisible, 465
Terminate, 254
Text, 465
TextHeight, 218
TextWidth, 218
Time, 132
Time( ), 77
Timer, 77; 255
TimeSerial( ), 77
TimeValue( ), 78
Title, 466
Top, 466; 506
ToPage, 467
TopIndex, 467
TopRow, 468
TrackDefault, 468
Transactions, 469
TRANSFORM, 503
Trim( ), 78
TwipPerPixelY, 470
TwipsPerPixelX, 469
Type, 133; 470
TypeName( ), 78

U

UBound, 472
UBound( ), 79
UCase( ), 79
Undatable, 473
Underline, 472
UNION, 509
Unique, 473
Unload, 134; 255
Unlock, 134
Update, 219; 504
UpdateControls, 220
Updated, 256
UpdateOptions, 474
UpdateRecord, 220
UseMnemonic, 475
UserName, 475

V

V1xNullBehavior, 476
Val( ), 79
Validate, 256
ValidateOnSet, 477
ValidationRule, 477
ValidationText, 479
Value, 479
Var( ), 495
Variant, 33
VarP( ), 496
VarType( ), 80
Verb, 481
Version, 481
Visible, 481
VisibleCols, 482
VisibleCount, 482
VisibleItem, 483
VisibleRows, 483

W

Weekday( ), 80
Weight, 483
WhatsThisButton, 484
WhatsThisHelp, 484
WhatsThisHelpID, 485
WhatsThisMode, 221
WHERE, 517
While...Wend, 135
Width, 486
Width #, 135
WindowList, 487
WindowState, 488
With, 135
WITH OWNERACCESS OPTION, 507
WordWrap, 488
Write#, 136

X

X1, 489
X2, 489

Y

Y1, 489
Y2, 490
Year( ), 81

Z

Zoom, 490
ZOrder, 221

≡1792

ISBN 963-577-176-2

HÍD A SZÁMÍTÁSTECHNIKÁHOZ....

az LSI

szak- és tankönyvei

80386/80486 II.	Dr. Kovács Magda	1062 Ft
A számítógépes grafika	Nagy-Perjés	892 Ft
Bevezetés a közigazgatási informatikába	Mezey Gyula-Mezey Gyuláné	490 Ft
Bevezetés az operációkutatásba	Hillier-Lieberman	2321 Ft
EXCEL 7 Windows alatt	Pétery Kristóf	1500 Ft
Fox Pro program gyakorlatok	Vágó Árpád	750 Ft
INTERNET CD melléklettel	Hargittai-Kaszanyiczki	1270 Ft
Mikroelektronikai értelmező szótár I-II.	Dr. Kovács Magda	1151 Ft
MULTIMÉDIA	Tóth Dezső	700 Ft
OS/2	Móricz Attila	1339 Ft
OS/2 Warp 3 magyar nyelvű változathoz	Móricz Attila	1446 Ft
PC Vírusok	Tóth Szabolcs	680 Ft
PENTIUM I.	Agárdi-Hadi	1071 Ft
PENTIUM II.	Agárdi-Hadi	1300 Ft
Sokoldalú szövegszerkesztő WORD 6.0 /a magyar nyelvű programhoz/	Kunos-Sőrés	1335 Ft
Sokoldalú szövegszerkesztő WORD 6.0 /az angol nyelvű programhoz/	Kunos-Sőrés	1335 Ft
Windows 95 Microsoft és NetWare hálózatokban	Móricz Attila	1490 Ft
WORDBASIC makrónyelv	Móricz Attila	1642 Ft
WORD 7.0 WINDOWS 95 alatt (magyar nyelvű változat)	Pétery Kristóf	1650 Ft
Egyszerűen a vállalkozásról	Dr. Pálinkás Jenő	1113 Ft
Útban a felsőbb matematikához	Dr. Kósa András	2400 Ft
Egy könyvben a középiskolai matematika	Dr. Korányi Erzsébet	1245 Ft
PowerPoint 7.0	Pétery Kristóf	1294 Ft

Áraink az ÁFÁ-T nem tartalmazzák

LSI Oktatóközpont Alapítvány
1037 Budapest, Bécsi út 324.
Tel: 250-60-13
Fax: 250-60-22